Programmation et indécidabilités dans les systèmes complexes

Nicolas Ollinger

Année 2008

HABILITATION À DIRIGER DES RECHERCHES

présentée à

L'UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

École Doctorale STIC

Spécialité : **Informatique**

par

Nicolas OLLINGER

Programmation et indécidabilités dans les systèmes complexes

Soutenue le 3 décembre 2008 devant la commission d'examen composée de :

M. Jean-Claude BERMOND	CNRS, Université de Nice-Sophia Antipolis et INRIA	
M. Olivier BOURNEZ	École Polytechnique	*Rapporteur*
M. Bruno DURAND	Université de Provence	
M. Enrico FORMENTI	Université de Nice-Sophia Antipolis	
M. Jarkko KARI	Université de Turku, Finlande	*Rapporteur*
M. Alexander SHEN	CNRS et Université de Provence	*Rapporteur*
Mme. Brigitte VALLÉE	CNRS et Université de Caen	*Présidente*

REMERCIEMENTS

Une habilitation à diriger des recherches se construit, tout autant qu'une thèse, sur des rencontres et des échanges. Je voudrais remercier ici toutes les personnes qui ont rendu possible l'écriture du présent mémoire et la soutenance de cette habilitation.

Soutenir une habilitation nécessite un jury. Je remercie Olivier Bournez, Jarkko Kari et Alexander Shen de m'avoir fait l'honneur de prendre part au jury et de rapporter sur le présent mémoire. Le premier pour sa lecture minutieuse du corps du document et son point de vue extérieur, source de nouveaux questionnements pertinents. Le second pour sa lecture minutieuse de l'ensemble des annexes et son point de vue de spécialiste incontournable, mais aussi pour sa grande hospitalité et ses échanges intéressants sur les automates cellulaires, la Finlande et les pavages — puisse notre collaboration se poursuivre dans les années à venir. Le troisième enfin pour son humilité, ses capacités à faire naître des questions très simples qui s'avèrent fondamentales, mais aussi pour le programme de recherche qu'il m'attribue dans son rapport.

Je remercie aussi Brigitte Vallée, la présidente du jury, ainsi que Jean-Claude Bermond et Enrico Formenti qui se sont joints aux précédents pour participer au jury, les deux premiers avec un regard très extérieur, le troisième comme spécialiste reconnu de la dynamique des automates cellulaires.

Enfin, je remercie tout particulièrement Bruno Durand non seulement pour avoir accepté d'être le septième membre de ce jury et pour ses contributions au bon déroulement de cette habilitation, mais surtout pour ces cinq riches années de collaboration où j'ai pu être témoin de sa capacité à porter des projets ambitieux, à résoudre les problèmes, à prendre des décisions courageuses tout en sachant rester à l'écoute.

Diriger des recherches, c'est aussi encadrer des étudiants. Je remercie les anciens doctorants Vincent Bernardi et Gaétan Richard dont j'ai eu le plaisir de co-encadrer et d'encadrer les travaux. Je remercie aussi les stagiaires de DEA/M2 et de 1re année de l'ÉNS Lyon Pierre Guillon, Florian Richoux, Alexandre Buisse et Timo Jolivet dont j'ai eu le plaisir d'encadrer les travaux. Je remercie plus particulièrement Gaétan Richard dont j'ai encadré le stage de 1re année en 2003 à l'ÉNS Lyon, qui m'a rejoint à Marseille pour son stage de M2 suivi de sa thèse qui s'achève en même temps que cette habilitation. C'est une riche expérience de transmettre la passion de ses thèmes de recherche et de voir émerger de cette interaction des résultats originaux. J'espère que nos routes se croiseront encore pour de fructueuses collaborations.

Chercher, c'est aussi participer à la vie d'une équipe et rencontrer d'autres chercheurs. Je remercie les membres de l'équipe Escape dont j'ai eu le plaisir de participer à la création et dont émerge une riche émulation. Je remercie plus particulièrement Grégory Lafitte qui est un des premiers à nous avoir rejoints et avec qui ce fût un plaisir d'organiser STACS 2006, JAC 2008 et autres FRAC.

Je remercie l'équipe MC3 de l'I3S qui m'a accueilli parmi ses membres associés et l'université de Nice-Sophia Antipolis où j'ai le plaisir de défendre cette habilitation. Je remercie plus particulièrement Enrico Formenti et Bruno Martin sans qui la gestion administrative et l'organisation pratique de la soutenance auraient relevé d'un vrai casse-tête loin de mon université marseillaise.

Enfin, je remercie l'ensemble de la communauté automates cellulaires, pavages et systèmes complexes discrets pour les interactions fructueuses à l'occasion des conférences et groupes de travail, comme le groupe francophone FRAC que nous organisons depuis 2006. Je remercie plus particulièrement ZAG ainsi que mes directeurs de thèse Marianne Delorme et Jacques Mazoyer qui m'ont transmis le goût de ces objets sur lesquels j'ai travaillé ces dix dernières années.

Ces travaux seraient bien plus difficiles à mener en dehors des institutions de recherche qui nous fournissent moyens humains, financiers et matériels. Je remercie le CNRS qui a cru en mon projet et m'a attribué un an et demi de délégation pour préparer cette habilitation.

Je ne saurais clore cette page sans remercier Alexandra pour sa présence, son attention et sa patience... Merci.

TABLE DES MATIÈRES

INTRODUCTION

« Nous ne sommes que des grains de sable
mais nous sommes ensemble.
Nous sommes comme les grains de sable
sur la plage, mais sans les grains de sable
la plage n'existerait pas. »

poème en langue Yamato

Le présent mémoire est une synthèse des activités de recherche que j'ai menées jusqu'à aujourd'hui. Il est constitué de deux parties. Le texte principal, qui comporte deux chapitres, présente mes différents thèmes de recherche, les résultats obtenus, les méthodes de preuve, ainsi que les définitions et propriétés nécessaires à leur compréhension. Ce texte n'est pas très technique et contient des renvois à la seconde partie du mémoire. Les annexes reproduisent une sélection d'articles, plus techniques, qui contiennent les démonstrations.

Un système complexe est un système constitué d'un ensemble d'entités qui interagissent localement, engendrant des comportements globaux, émergeant du système, qu'on ne sait pas expliquer à partir du comportement local, connu, des entités qui le constituent. Discipline nouvelle, s'intéressant à une problématique transverse à d'autres disciplines établies, l'étude des systèmes complexes est un domaine pluridisciplinaire émergent. Les travaux présentés dans le présent mémoire ont pour objet de mieux cerner les liens entre certaines propriétés des systèmes complexes et le calcul. Par calcul, il faut entendre l'objet d'étude de l'informatique, c'est-à-dire, si on met de côté les problèmes d'entrée/sortie, le déplacement et la combinaison d'informations. À l'aide d'outils issus de l'informatique, l'algorithmique et la programmation dans les systèmes complexes sont abordées selon trois points de vue. Une première forme de programmation, dite externe, consiste à développer l'algorithmique qui permet de simuler les systèmes étudiés. Une seconde forme de programmation, dite interne, consiste à développer l'algorithmique propre à ces systèmes, qui permet de construire des représentants de ces systèmes qui exhibent des comportements programmés. Enfin, une troisième forme de programmation, de réduction, consiste à plonger des propriétés calculatoires complexes dans les représentants de ces systèmes pour établir des résultats d'indécidabilité — indice d'une grande complexité calculatoire qui participe à l'explication de la complexité émergente.

Afin de mener à bien cette étude, les systèmes complexes sont modélisés par des automates cellulaires. Les automates cellulaires constituent un modèle de système complexe très régulier et synchrone constitué d'une grille infinie régulière de cellules à état fini qui interagissent selon une règle déterministe locale. Le modèle des automates cellulaires, simple mais pas simpliste, offre une dualité pertinente pour établir des liens entre complexité des propriétés globales et calcul. En effet, un automate cellulaire peut être décrit à la fois comme un réseau d'automates, offrant un point de vue familier de l'informatique, et comme un système dynamique discret, une fonction définie sur un espace topologique, offrant un point de vue familier de l'étude des systèmes dynamiques discrets.

Une première partie de nos travaux, exposés dans le chapitre 1, concerne l'étude de l'objet automate cellulaire proprement dit. L'observation expérimentale des automates cellulaires distingue, dans la littérature, deux formes de dynamiques complexes dominantes. D'une part, certains automates cellulaires présentent des comportements de type chaotique, mélangeant toute l'information présente dans l'espace et le temps. Cette première forme de complexité, très étudiée par les dynamiciens et les topologues, ne fait pas l'objet du présent mémoire. D'autre part, certains automates cellulaires présentent une dynamique dans laquelle émergent des structures simples, de sortes de particules qui évoluent dans un domaine régulier, se rencontrent lors de brèves collisions, avant de générer d'autres particules. Cette seconde forme de complexité, dans laquelle transparaît une notion de quanta d'information localisée en interaction, est l'objet de nos études. Un premier champ de nos investigations est d'établir une classification algébrique, le groupage, qui tend à rendre compte de ce type de comportement. Cette classification met à jour un type d'automate cellulaire particulier : les automates cellulaires intrinsèquement universels. Un automate cellulaire intrinsèquement universel est capable de simuler le comportement de tout automate cellulaire. C'est l'objet de notre second champ d'investigation. Nous caractérisons cette propriété et démontrons son indécidabilité. Enfin, un troisième champ d'investigation concerne l'algorithmique des automates cellulaires à particules et collisions. Étant donné un tel automate cellulaire, un ensemble de particules et de collisions, nous étudions l'ensemble des interactions possibles et proposons des outils pour une meilleure programmation interne à l'aide de ces collisions.

Une seconde partie de nos travaux, exposés dans le chapitre 2, concerne la programmation par réduction. Afin de démontrer l'indécidabilité de propriétés dynamiques des automates cellulaires, nous étudions d'une part les problèmes de pavage du plan par des jeux de tuiles finis et d'autre part les problèmes de mortalité et de périodicité dans les systèmes dynamiques discrets à fonction partielle. Cette étude nous amène à considérer des objets qui possèdent la même dualité entre description combinatoire et topologique que les automates cellulaires. Une notion d'apériodicité joue un rôle central dans l'indécidabilité des propriétés de ces objets.

L'étude des liens entre systèmes complexes et calcul n'en est qu'à ses débuts. Il reste de nombreuses voies à explorer, des outils à exhiber, des approches à unifier. Les automates cellulaires constituent un modèle agréable à étudier, mais les résultats obtenus offrent une information qualitative sur le calcul dans les systèmes complexes en général. Ce mémoire est une contribution à l'étude primitive des liens entre dynamique et calcul.

Nota Bene Ce mémoire est un résumé de mes activités de recherche. Par nature, il est donc incomplet, certains thèmes de recherche secondaires comme la combinatoire des mots [J1] ou les automates cellulaires conservatifs [P1] ont été volontairement omis. De même, il ne s'agit pas d'un état de l'art, ni même d'un compendium, sur les automates cellulaires, la calculabilité ou encore les pavages ; les aspects développés l'étant pour mettre en perspective les travaux effectués. Toutefois, lorsqu'ils existent, des états de l'art pertinents sont proposés au fil du texte.

CHAPITRE 1

AUTOMATES CELLULAIRES, GÉOMÉTRIE ET CALCUL

« This model is based on a crystalline medium; we will be able to construct it in two dimensions and to use the quadratic (regular) lattice. Each lattice point of the crystal will be able to assume a finite number of different state (say N states) and its behavior will be described (or controlled) by an unambiguous transition rule, covering all transitions between these states, as affected by the states of the immediate neighbors. »

John von Neumann, in *Theory of Self-Reproducing Automata* [74, p. 132]

1.1 Automates cellulaires

Les automates cellulaires, modèle simple et formel de système complexe, constituent l'objet d'étude principal de nos travaux. Cet objet étant par nature dual et nos travaux se nourrissant de cette dualité, deux façons *a priori* très différentes s'offrent à nous pour le définir. Nous les présentons toutes deux successivement, en commençant par l'approche la plus classique en informatique fondamentale.

Un automate cellulaire est constitué d'une collection de petites machines élémentaires à état fini disposées selon une grille régulière. Ces machines élémentaires, toutes identiques mais potentiellement dans des états distincts, changent d'état de manière synchrone et uniforme : chaque machine observe l'état de ses voisines selon un même voisinage afin de déterminer, par sa règle de transition, son nouvel état. Formellement, un *automate cellulaire* est la donnée d'un quadruplet (d, S, V, f) constitué d'une grille régulière $\mathbb{Z}^d$, où d est la *dimension de l'automate*, d'un *ensemble fini d'états* S, d'un *voisinage* $V \subseteq_{\text{fini}} \mathbb{Z}^d$ et d'une *règle de transition* $f : S^V \to S$. Une *configuration* $c \in S^{\mathbb{Z}^d}$ d'un tel automate est un coloriage de la grille par les états. La *fonction globale* F de l'automate associe à une configuration la configuration suivante en appliquant uniformément la règle de transition en translatant le voisinage en tout point de la grille : pour toute configuration $c \in S^{\mathbb{Z}^d}$ et tout point $z \in \mathbb{Z}^d$ de la grille, $F(c)(z) = f(c_{|z+V})$. Le *diagramme espace-temps* $\Delta(c) \in S^{\mathbb{N}\times\mathbb{Z}^d}$ engendré par une configuration $c \in S^{\mathbb{Z}^d}$ est la suite des itérés de F : $(c, F(c), F^2(c), \ldots)$ où $F^{k+1} = F \circ F^k$.

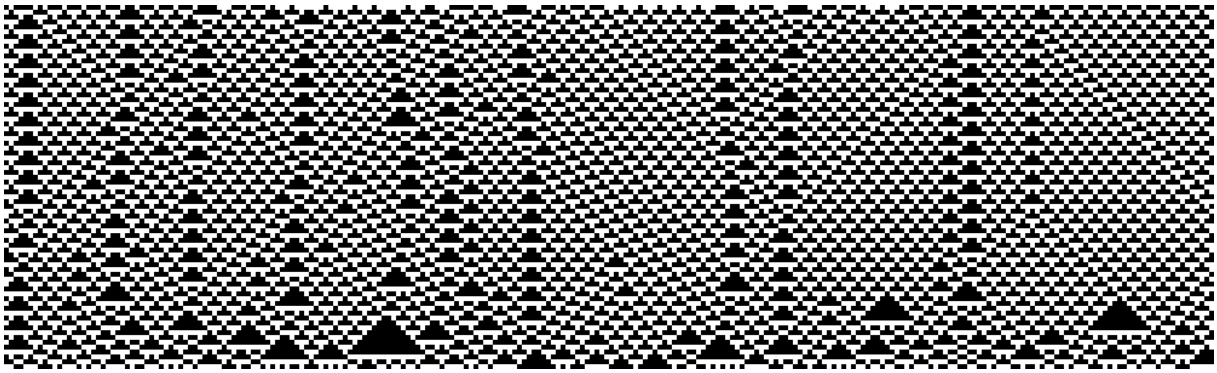


FIGURE 1.1 – diagramme espace-temps de la règle 54

Exemple 1. L'automate cellulaire $(1, \{0,1\}, \{-1,0,1\}, f_{54})$ dont la règle de transition satisfait $f_{54}(x,y,z) = \lfloor 54/2^{4x+2y+z} \rfloor \pmod 2$ engendre le diagramme espace-temps représenté sur la figure 1.1 où la flèche du temps oriente le diagramme de bas en haut. ◇

```
open Graphics

let w = 250 and h = 75

let c = Array.create w 0

let init () =
        let v =ref 1 in
        for i = 0 to w − 1 do
                c.(i) ← !v mod 2;
                v := (75 × !v) mod 65537
        done

let draw y =
        for x = 0 to w − 1 do
                if c.(x) ≡ 1 then plot x y
        done

let f x y z = (54 lsr (4 × x + 2 × y + z)) land 1

let next () =
        let l =ref c.(w − 1)
        and r = c.(0) in
        for i = 0 to w − 2 do
                let v = c.(i) in
                c.(i) ← f !l c.(i) c.(i + 1);
                l := v
        done;
        c.(w − 1) ← f !l c.(w − 1) r

let _ =
        open_graph (Printf.sprintf "␣%ix%i" w h);
        init ();
        for y = 0 to h − 1 do
                draw y;
                next ()
        done;
        read_key ()
```

Src. 1.1 – générateur de diagrammes espace-temps pour la règle 54

Cette définition incite à écrire un petit programme dans le style du source 1.1 pour générer des diagrammes espace-temps et s'approprier l'objet. Pourtant, cette section aurait pu commencer tout autrement avec un point de vue *systèmes dynamiques*, comme suit.

Un *système dynamique discret* est une paire (X, F) constituée d'un *espace topologique* X sur lequel est itérée une *application continue* $F : X \to X$. Une *configuration* est un point $c \in X$. L'*orbite* d'une configuration $c \in X$ est l'ensemble des configurations obtenues en itérant F : $\mathcal{O}_F(c) = \{c, F(c), F^2(c), \ldots\}$. L'*espace de Cantor* de dimension d associé à l'alphabet fini S est l'espace $S^{\mathbb{Z}^d}$ muni de la topologie produit de la topologie discrète sur S. Cet espace est métrique, compact et parfait. La $i$ème *translation élémentaire* de $S^{\mathbb{Z}^d}$ est l'application $\sigma_i : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ vérifiant, pour tout point $(x_1, \ldots, x_d) \in \mathbb{Z}^d$ et toute configuration $c \in S^{\mathbb{Z}^d}$, $\sigma_i(c)(x_1, \ldots, x_i + 1, \ldots, x_d) = c(x_1, \ldots, x_i, \ldots, x_d)$. Un *automate cellulaire* est un système dynamique discret $(S^{\mathbb{Z}^d}, F)$ où F est un endomorphisme de $(S^{\mathbb{Z}^d}, \sigma_1, \ldots, \sigma_d)$, *i.e.* une application continue $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ qui commute avec les translations.

L'approche est très différente, insistant sur les propriétés globales des automates cellulaires, occultant le caractère local et fini. Avant de préciser nos contributions à l'étude des automates cellulaires, à partir de la section 1.2, nous invitons le lecteur à une escapade, un florilège dans l'étude des automates cellulaires. Le lecteur profane saura tirer parti de [19, 23, 50] ou encore [T1] pour compléter ce qui suit.

1.1.1 Expérimentation, algorithmique et complexité

C'est sur les conseils de S. Ulam qu'à la fin des années 40 J. von Neumann introduit les automates cellulaires. Il s'agit alors de choisir un modèle raisonnable du monde biologique qui permette de coder aisément des automates auto-reproducteurs [74], des machines

obtenues comme point fixe de l'association d'une machine universelle pour le calcul et d'une machine universelle pour la construction (voir [74, pp. 85–87]). Dans les années 60, K. Zuse propose d'utiliser les automates cellulaires comme modèle de la physique, en particulier en physique des particules [97]. Les automates cellulaires, par leur uniformité et leur localité, constituent un outil de modélisation *rudimentaire* permettant une approche expérimentale naïve facile à mettre en oeuvre : on commence par fixer une grille et une discrétisation des états, on identifie la règle d'interaction locale, on peut alors simuler l'automate cellulaire, observer des diagrammes espace-temps, effectuer des traitements statistiques sur les données obtenues, etc. Une simple recherche bibliographique à partir du mot clef *cellular automata* met en avant une profusion de modélisations utilisant les automates cellulaires comme premier modèle, voir parfois comme unique modèle, faute de modèle continu *ad hoc*, et dans des domaines aussi divers que la chimie, la biologie, la géographie ou encore la modélisation de trafics routiers.

Cependant, une modélisation efficace à l'aide d'automates cellulaires est loin d'être triviale. Une fois franchi l'hydre de la discrétisation et identifiée une règle locale compatible avec cette discrétisation, pour peu qu'elle existe, il convient d'exhiber les configurations qui ont un sens et de simuler l'automate cellulaire pendant un temps suffisant. Enfin, comme nous le verrons dans la suite, il est généralement extrêmement difficile d'extraire des propriétés globales à partir des règles locales et l'utilisation du modèle restera le plus souvent purement expérimentale. On notera cependant quelques succès, telle la modélisation de fluides par des automates cellulaires décrivant des collisions de particules. Particules dont il est démontré directement que leurs interactions vérifient, comme le modèle continu, les équations de Navier-Stokes [93], faisant à la fois le lien d'une part entre comportement local et comportement global et d'autre part entre le modèle discret et son analogue continu modélisé par des équations différentielles.

Automates cellulaires et simulation

L'informatique joue ici un rôle des plus classiques, *extrinsèque*, apportant son lot de questions et d'outils pour la représentation des objets, les langages de description de règles de transition, l'algorithmique de la simulation, les questions de complexité associées, etc.

Configurations L'ensemble des configurations n'étant pas dénombrable, il est nécessaire de considérer uniquement certaines configurations récursives. L'ensemble des configurations récursives amenant, par le théorème de Rice, son cortège de propriétés indécidables, on trouve dans la littératures principalement deux types de configurations très régulières et facilement manipulables.

En dimension d, une *configuration périodique* est une configuration admettant une base de d vecteurs de périodicité non colinéaires. Les automates cellulaires étant définis localement et uniformément, ils préservent les symétries et l'image d'une configuration périodique admet les mêmes périodes. Cet avantage pour la simulation est aussi un inconvénient : l'orbite d'une configuration périodique définie par un motif de n cellules est ultimement périodique, de transitoire et de période de longueurs au plus $|S|^n$.

Une *configuration s-finie* est une configuration partout égale à $s \in S$ sauf sur un support fini. L'image d'une configuration finie est une configuration finie, éventuellement de

support plus grand. C'est le second type de configuration le plus courant dans la littérature. La notion de configuration finie se généralise aisément en la notion de configuration ultimement périodique, nous discuterons dans la section 1.4 de l'extension de cette notion à la dimension 2.

Règle de transition Dans le programme naïf du source 1.1, la règle de transition est donnée par son index parmi les 256 automates cellulaires de même voisinage et ensemble d'états. Cette approche ne passe pas à l'échelle : avec n états et un voisinage de cardinal m, le nombre de règles distinctes est n^{n^m}. Les règles de transition sont donc rarement spécifiées en extension.

Simulation Dans le programme naïf du source 1.1, la règle de transition est appliquée pour chaque cellule en tout temps. Ainsi, la simulation pendant n étapes sur une configuration périodique de taille n^d requiert n^{d+1} étapes de calcul. Bien que la simulation soit massivement parallélisable, elle devient difficilement traitable quand n atteint le million de cellules et qu'il s'agit de traiter des milliers de configurations. Peut-on faire mieux ?

Considérons le problème de décision suivant : étant donné un automate cellulaire, un motif de $(2n+1)^d$ états et un état distingué s, l'image itérée n fois du motif par l'automate est-elle égale à s ? Le lecteur se convaincra aisément[1] que ce problème est P-complet pour la réduction *many-one* en espace logarithmique.

Si on ne peut pas faire mieux que l'approche naïve dans le cas général, on peut espérer une accélération lorsque les diagrammes espace-temps sont très réguliers ou lorsque l'automate cellulaire appartient à une famille d'automates aux propriétés algébriques très fortes. Ainsi, une approche type diviser pour régner permet à R. Gosper [36] de simuler efficacement, avec son algorithme HashLife, le célèbre *jeu de la vie* de J. H. Conway [10], qui possède des diagrammes espace-temps très creux. Cette approche reste pertinente pour le traitement et le stockage des diagrammes espace-temps des automates cellulaires à particules de la section 1.4.

Exploration expérimentale

Cette approche expérimentale, par simulation, des automates cellulaires a aussi nourri l'étude des propriétés de l'objet. Ainsi, dans les années 80, S. Wolfram [94] établit, à partir d'une observation approfondie des diagrammes espace-temps des 256 automates cellulaires de dimension 1 les plus simples (représentés sur la figure 1.2), sa célèbre classification expérimentale :

> « [...] Indeed, among all kinds of cellular automata, it seems that the patterns which arise can almost always be assigned quite easily to one of just four basic classes [...]
>
> These classes are conveniently numbered in order of increasing complexity, and each one has certain immediate distinctive features.
>
> In class 1, the behavior is very simple, and almost all initial conditions lead to exactly the same uniform final state.
>
> In class 2, there are many different possible final states, but all of them consist just of a certain set of simple structures that either remain the same forever or repeat every few steps.
>
> In class 3, the behavior is more complicated, and seems in many respects random, although triangles and other small-scale structures are essentially always at some level seen.
>
> And finally [...] class 4 involves a mixture of order and randomness: localized structures are produced which on their own are fairly simple, but these structures move around and interact with each other in very complicated ways. [...] »
>
> *S. Wolfram [95, chapitre 6, pp. 231–235]*

[1] il s'agit d'adapter la preuve classique de la P-complétude de Circuit-Value due à R. E. Ladner [55].

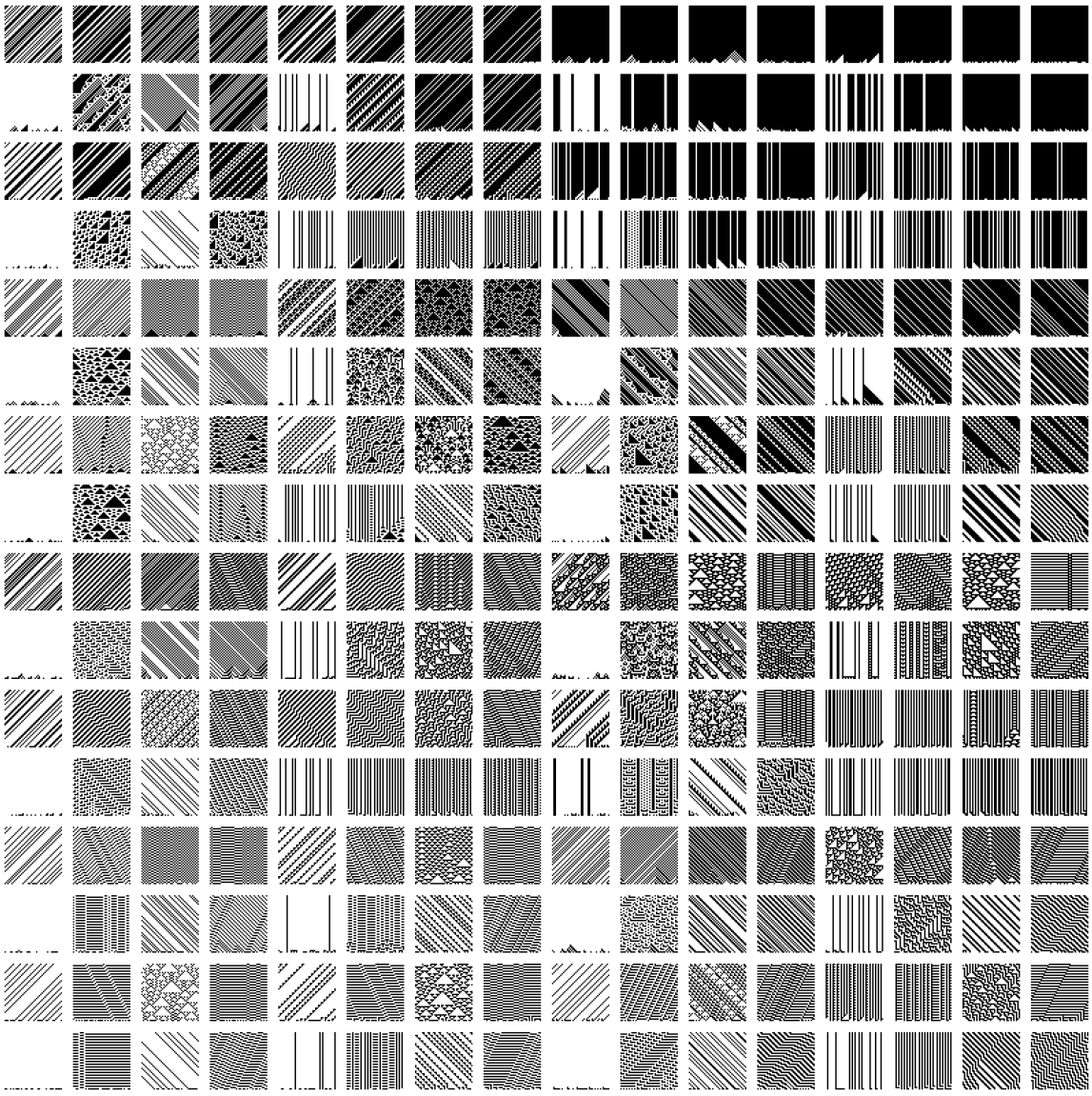


FIGURE 1.2 – diagrammes espace-temps des 256 automates cellulaires élémentaires

Si cette tentative informelle de classification reste très dépendante de l'observateur et se révèle informalisable, voir J. T. Baldwin et S. Shelah [4], elle présente un double intérêt. D'une part, elle présente sous une forme très élémentaire le *motto* des systèmes complexes : un grand nombre d'objets très simples en interaction locale peuvent engendrer des comportements globaux très complexes. D'autre part, elle met en avant quatre comportements types des automates cellulaires : deux types de dynamiques qualifiées de simples dont on verra au chapitre 2 qu'elles sont indécidables et deux types de dynamiques qualifiées de complexes. La première forme de complexité correspond au chaos déterministe des automates cellulaires qui distribuent et mélangent au maximum toute l'information contenues dans leurs configurations, voir E. Formenti [33]. La seconde forme de complexité semble plus algorithmique : des structures simples s'auto-organisent, générant des diagrammes espace-temps très réguliers, où des particules déplacent l'information régulièrement et la combinent lors de collisions ponctuelles. Cette seconde forme de complexité motive les section 1.2 et 1.4.

1.1.2 Construction et programmation

Le caractère discret et synchrone des automates cellulaires les rend propices au calcul : on peut construire des automates cellulaires ayant des comportements *ad hoc* sur des ensembles de *configurations codantes*. Ce faisant, on déplace l'objet d'étude de l'informatique à l'intérieur du modèle. Ainsi, on construit simplement un automate cellulaire *simulant* le comportement d'une machine de Turing : certains états codent les lettres du ruban, d'autres états codent la tête de calcul de la machine. On considère les configurations constituées d'une unique tête de calcul disposée sur un unique ruban de calcul biinfini. Étant donné un automate cellulaire, une configuration finie et un motif d'états, savoir si le motif apparaît dans l'orbite de la configuration est indécidable, par réduction du problème de l'arrêt des machines de Turing.

Les premiers automates cellulaires considérés, par J. von Neumann [74] puis par E. F. Codd [15], E. R. Banks [6], C. G. Langton [56], entre autres, sont des automates cellulaires de dimension 2 dont les règles de transition sont spécifiées de sorte à simuler des signaux se déplaçant dans des gaines, des croisements de fils, des portes logiques agissant sur ces signaux et des organes activés par ces signaux pour construire ou détruire de l'information. La capacité à simuler des circuits booléens assure l'universalité pour le calcul et en y combinant les organes de construction on obtient simplement l'auto-reproduction comme un point fixe, c'est-à-dire un automate universel pour la construction construisant une copie de lui-même. Les différentes formes d'universalité qui s'y rapportent sont discutées dans la section 1.3.

Simuler des machines séquentielles avec des automates cellulaires n'est pas très satisfaisant. De fait, motivées par l'étude du parallélisme pour lequel les automates cellulaires offrent un modèle massivement parallèle, se sont développés une algorithmique et des techniques de programmation parallèle qui leur sont propres, que nous ne détaillerons pas ici. Les constructions sont alors valides dès la dimension 1 et sont de nature très géométrique. Ainsi le problème de synchronisation d'une ligne de fusiliers introduit par E. F. Moore [70] consiste à synchroniser un segment de cellules en temps optimal. La meilleure solution connue à ce jour, due à J. Mazoyer [62], exploite une construction géométrique récursive. P. C. Fischer [32] propose une construction géométrique permettant à une cellule issue d'une configuration finie de savoir à tout temps t si t est un nombre premier. De nombreux travaux sont menés pour étudier la reconnaissance de langages par automates cellulaires, voir l'article fondateur d'A. R. Smith III [87] ou encore la synthèse de M. Delorme et J. Mazoyer [24]. Cette algorithmique se caractérise par les outils suivants : signaux, synchronisation, calcul géométrique.

Dans la suite, afin d'étudier la dynamique des automates cellulaires, que ce soit pour construire des exemples ou réduire des problèmes entre eux, nous aurons besoin de programmer des automates cellulaires particuliers en considérant leur évolution en partant de toutes les configurations. Dans ce contexte, les outils algorithmique cités ci-dessus sont mis à contribution (cf. section 1.3 et chapitre 2).

1.1.3 Topologie apocryphe de l'espace de Cantor

La topologie de Cantor est un outil qui apparaît très tôt dans l'étude des automates cellulaires et qui se révèle très pratique pour démontrer de nombreux résultats, comme nous le

verrons dans la suite de ce chapitre mais aussi au chapitre 2 lorsque nous considérerons des machines de Turing et des pavages du plan. Il s'agit, sur $S^{\mathbb{Z}^d}$, de la topologie produit de la topologie discrète sur S. Cette topologie est métrique, compacte et parfaite. La distance entre deux configurations c et c' est définie, à un facteur multiplicatif de normalisation près, par :

$$d(c,c') = 2^{-\min\left\{\|z\|_\infty \middle| c(z)\neq c'(z)\right\}} \quad .$$

Cependant, le lecteur ne doit pas se laisser abuser par le caractère topologique de cet outil. S'il elle permet d'analyser les automates cellulaires à la lumière des outils de la dynamique topologique, la topologie de Cantor est de nature essentiellement combinatoire. Les principales propriétés topologiques peuvent s'obtenir par extraction, à l'aide du lemme de König.

Lemme 1 (König)**.** *Tout arbre infini à branchement fini possède une branche infinie.*

Un *motif* $m \in S^{\sup(m)}$ est une configuration partielle de support fini $\sup(m) \subseteq_{\text{fini}} \mathbb{Z}^d$. Un motif m est un *sous-motif* d'un motif m', noté $m \prec m'$, si $m'_{|\sup(m)} = m$. Le translaté de vecteur $u \in \mathbb{Z}^d$ d'un motif m est le motif m' de support $u+\sup(m)$ défini en tout point $z \in \sup(m)$ par $m'(z+u) = m(z)$. Un motif m *apparaît* dans une configuration $c \in S^{\mathbb{Z}^d}$ si un translaté $\sigma_u(m)$ de m est un sous-motif de c : $\sigma_u(m) \prec c$. Un *motif élémentaire*, de taille $n \in \mathbb{N}$, est un motif de support $[-n,n]^d$. Le *cylindre* $[m] \subseteq S^{\mathbb{Z}^d}$ de motif m est l'ensemble des configurations contenant le motif m en leur centre :

$$[m] = \left\{ c \in S^{\mathbb{Z}^d} \middle| c_{|\sup(m)} = m \right\} \quad .$$

Un *cylindre élémentaire* est un cylindre de motif élémentaire. Un cylindre $[m]$ est un sous-cylindre d'un cylindre $[m']$, noté $[m] \prec [m']$, si m est un sous-motif de m'. L'ensemble des cylindres élémentaires est noté $\mathcal{E}$. L'arbre de König d'un ensemble de configurations est l'arbre de ses cylindres élémentaires ordonnés par la relation sous-cylindre. Plus formellement, l'*arbre de König* $\mathcal{A}_C$ d'un ensemble de configurations $C \subseteq S^{\mathbb{Z}^d}$ est l'arbre à branchement fini (V_C, E_C) où

$$\begin{aligned} V_C &= \{[m] \in \mathcal{E} \mid C \cap [m] \neq \emptyset\} \quad \text{et} \\ E_C &= \left\{ ([m],[m']) \in V_C^2 \;\middle|\; [m] \prec [m'] \;\wedge\; \forall [m''] \in V_C \; \neg([m] \prec [m''] \prec [m']) \right\} \quad . \end{aligned}$$

L'arbre de König d'un ensemble de configurations non vide est un arbre infini à branchement infini. Chaque branche infinie de l'arbre identifie une configuration unique : l'intersection des cylindres qui composent la branche. La *ramure* de l'arbre de König $\mathcal{A}_C$ est l'ensemble $\overline{\mathcal{A}_C}$ des configurations associées aux branches infinies.

Définition 1. La *topologie de König* de l'espace $S^{\mathbb{Z}^d}$ est définie par ses fermés : les ramures des arbres de König.

La ramure de l'arbre de König d'une ramure est égale à cette ramure. Le complémentaire de la ramure d'un arbre de König est exactement l'union des cylindres élémentaires qui ne sont pas des sommets de l'arbre. Comme l'ensemble des cylindres élémentaires forme une base dénombrable d'ouverts de l'espace de Cantor, les topologies de Cantor et de König coïncident.

Les propriétés topologiques se visualisent alors comme des opérations sur les arbres. Ainsi, l'adhérence d'un ensemble de configurations est la ramure de son arbre de König

et un ensemble de configurations est dense s'il intersecte tout cylindre élémentaire. De même, les ramures sont compactes car, d'un recouvrement d'une ramure par une famille infinie de cylindres élémentaires n'admettant pas de sous-recouvrement fini, le lemme de König extrait un point de la ramure qui n'est pas recouvert. Un fermé est d'intérieur non vide si sa ramure contient un cylindre élémentaire, c'est-à-dire si le sous-arbre enraciné en ce cylindre est complet. Le théorème de Baire s'obtient avec le lemme de König en lieu et place de l'axiome du choix : partant de l'hypothèse qu'une famille dénombrable de fermés d'intérieurs vides possède une union d'intérieur non vide, c'est-à-dire contenant un cylindre élémentaire, on réfute l'hypothèse en construisant récursivement une branche infinie dans ce cylindre qui n'appartient à aucun des fermés de la famille.

Par compacité, les *ourmés*[2] sont les unions finies de cylindres élémentaires. Dans la topologie de König, continuité signifie localité : pour connaître l'image d'une configuration sur un motif, il suffit d'observer un motif de la configuration initiale car l'image réciproque par une application continue d'un ourmé est un ourmé. En considérant les cylindres de motifs élémentaires de taille 1 et en ajoutant l'uniformité, c'est-à-dire l'invariance par translation, on obtient le théorème de Curtis-Hedlund-Lyndon qui caractérise les applications uniformes et locales :

Théorème 1 (G. A. Hedlund [41], D. Richardson [83])**.** *Les fonctions globales des automates cellulaires sont les applications continues qui commutent avec les translations.*

Une fois ce résultat acquis, on peut manipuler directement les fonctions globales des automates cellulaires, les composer, les inverser lorsqu'elles sont bijectives, ou encore en faire le produit cartésien : le résultat de ces opérations caractérise toujours des automates cellulaires. Le précédent théorème établit aussi un lien entre les automates cellulaires et la dynamique symbolique dont l'objet d'étude est le *sous-shift* : un ensemble de configurations fermé et clos par translation, voir l'ouvrage introductif de D. Lindt et B. Marcus [57].

1.1.4 Propriétés globales immédiates

Pour mieux simuler et construire des automates cellulaires, ou pour les étudier en tant que systèmes complexes, il convient d'étudier les propriétés de ces objets et plus précisément le lien entre les propriétés locales de la règle de transition f et les propriétés globales de la fonction globale F. Les propriétés les plus étudiées, depuis les années 60, et les mieux comprises sont les propriétés immédiates : celles qui ne dépendent pas de l'itération de la fonction globale.

Un automate cellulaire est *injectif* (resp. *surjectif*, *bijectif*) si sa fonction globale F est *injective* (resp. *surjective*, *bijective*). Un automate cellulaire est *réversible* s'il est bijectif et si l'inverse de sa fonction globale est la fonction globale d'un automate cellulaire. Une configuration d'un automate cellulaire est un *jardin d'Eden* si elle n'admet pas d'antécédent. Un motif est un *mot d'Eden* si toute configuration qui le contient est un jardin d'Eden. Un automate cellulaire est surjectif si et seulement si il n'admet aucun jardin d'Eden. Par compacité, un automate cellulaire est réversible si et seulement si il est bijectif et une configuration est un jardin d'Eden si et seulement si elle contient un mot d'Eden.

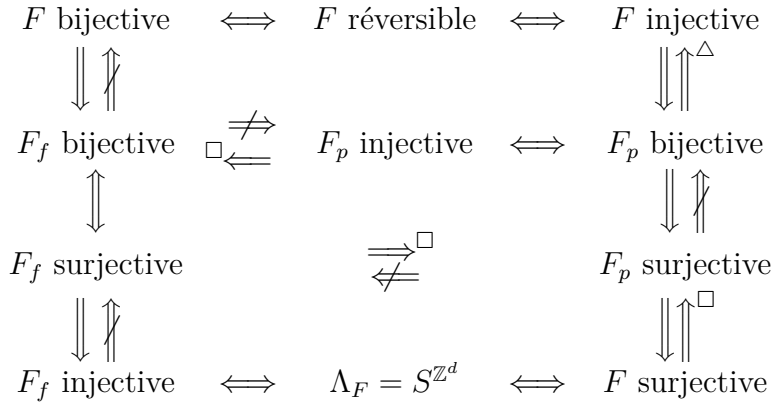
[2] ensembles à la fois ouverts et fermés.

La restriction de la fonction globale F aux configurations finies (resp. périodiques) est notée F_f (resp. F_p). On définit naturellement l'injectivité, la surjectivité et la bijectivité sur les configurations finies (resp. périodiques) en remplaçant F par F_f (resp. F_p). L'injectivité implique l'injectivité sur les configurations finies et périodiques. Par compacité et densité des configurations finies (resp. périodiques), la surjectivité sur les configurations finies (resp. périodiques) implique la surjectivité. Par un argument de comptage, les automates cellulaires préservant les symétries (et donc les périodes), un automate cellulaire injectif sur les configurations périodiques est aussi surjectif sur les configurations périodiques. Par compacité et par l'absurde, un automate cellulaire injectif est surjectif sur les configurations finies. Enfin, un résultat non trivial est obtenu par un argument combinatoire, il s'agit du théorème des jardins d'Eden, un des plus anciens résultats établis sur les automates cellulaires :

Théorème 2 (E. F. Moore [69] et J. Myhill [72])**.** *Un automate cellulaire est surjectif si et seulement si il est injectif sur les configurations finies.*

En dimension 1, les automates cellulaires agissent sur les configurations, des mots biinfinis, comme des transducteurs. Tout automate cellulaire dont le voisinage est inclus dans un segment de longueur k peut être représenté par un automate fini dont les sommets correspondent aux $k-1$ dernières lettres lues et dont les arcs associent à une lettre lue la lettre écrite sur la configuration image. Dans cet automate, une tranformation d'une configuration par F est un chemin biinfini duquel on extrait facilement des chemins périodiques. Ainsi, en dimension 1, un automate cellulaire injectif sur les configurations périodiques est injectif. De même, en dimension 1, un automate cellulaire surjectif est surjectif sur les configurations périodiques.

En enrichissant les résultats ci-dessus par des contre-exemples appropriés, dont on trouvera le détail dans [50], on peut résumer l'ensemble des connaissances par la figure 1.3 (librement inspirée de la thèse de B. Durand [25]). Sur cette figure, Λ_F dénote l'ensemble limite introduit dans la section 1.1.5.



△ signifie que la relation est vérifiée en dimension un, mais ne l'est pas pour les dimensions supérieures ; □ signifie que la relation est vérifiée en dimension un, la question est ouverte en dimension deux.

FIGURE 1.3 – Propriétés de la règle globale de transition et familles de configurations

Problème ouvert 1. *En dimension 2, lesquelles de ces trois implications sont vraies :*

(i) F_p *injective implique* F_f *surjective ?*

(ii) F_f *surjective implique* F_p *surjective ?*

(iii) F_f *injective implique* F_p *surjective ?*

La dissymétrie de traitement entre les dimensions un et supérieures tient principalement à l'absence de notion équivalente à la notion de langage régulier en dimensions supérieures. Ainsi, une étude attentive du transducteur introduit ci-dessus permet d'obtenir le résultat suivant :

Théorème 3 (S. Amoroso et Y. N. Patt [3])**.** *L'injectivité et la surjectivité des automates cellulaires de dimension un sont décidables.*

Par contre, comme nous le verrons plus en détail dans le chapitre 2, les choses se compliquent à partir de la dimension 2 :

Théorème 4 (J. Kari [48])**.** *L'injectivité et la surjectivité des automates cellulaires de dimension deux et supérieures sont indécidables.*

Pour en savoir plus sur les propriétés immédiates, le lecteur pourra consulter, entre autres, les travaux de G. A. Hedlund [41] et d'A. Maruoka et M. Kimura [60].

1.1.5 Dynamique et classifications

Comprendre la dynamique des automates cellulaires, c'est comprendre les liens entre les propriétés locales de la règle de transition f et les propriétés globales observées en itérant la fonction globale F. L'espace des phases d'un système dynamique discret est le graphe dirigé dont les sommets sont les configurations et dans lequel une arête relie chaque configuration à son image. Si les résultats d'indécidabilité évoqués plus haut rendent vaine toute tentative de description de l'espace des phases d'un automate cellulaire dans le cas général, cela reste possible lorsque la règle locale et son application sont très régulières d'un point de vue algébrique, par exemple lorsque l'ensemble d'états est un groupe dont la règle de transition est un morphisme, voir O. Martin *et al* [59].

Un outil de l'étude des systèmes dynamiques est la notion d'attracteur qui tend à décrire le comportement *typique* du système. Dans l'étude des automates cellulaires, ce rôle est classiquement joué par l'ensemble limite. Le lecteur consultera K. Čulik II *et al* [19, 20] pour plus d'informations sur les ensembles limites et leurs propriétés.

L'ensemble limite d'un automate cellulaire est constitué des configurations qui peuvent apparaître après n'importe quel temps dans un diagramme espace-temps, c'est le plus grand ensemble pour l'inclusion sur lequel la fonction globale est surjective. Formellement, l'*ensemble limite* Λ_F d'un automate cellulaire de fonction globale F et d'ensemble de configurations $S^{\mathbb{Z}^d}$ est le sous-shift non vide suivant, où F^0 est l'identité :

$$\Lambda_F = \bigcap_{t \in \mathbb{N}} F^t \left(S^{\mathbb{Z}^d} \right) \quad .$$

Si la compacité nous apprend que l'ensemble limite, limite d'une suite décroissante de fermés non vides, est un fermé non vide, elle permet aussi de se convaincre qu'il est exactement l'ensemble des configurations qui apparaissent dans les diagrammes espace-temps à temps biinfini, *i.e.* les configurations $c \in S^{\mathbb{Z}^d}$ pour lesquelles il existe $\Delta \in S^{\mathbb{Z}\times\mathbb{Z}^d}$ tel que $\Delta(0) = c$ et pour tout $t \in \mathbb{Z}$, $\Delta(t+1) = F(\Delta(t))$. Nous verrons au chapitre 2 que ces diagrammes biinfinis sont une variété particulière de pavages du plan qui lie automates cellulaires et modèles de calcul. Ces liens sont la cause du théorème qui suit.

L'ensemble limite d'un automate cellulaire contient toujours une configuration monochromatique, c'est-à-dire une configuration uniformément égale à un même état. Un automate cellulaire est nilpotent si son ensemble limite est un singleton. Si un automate cellulaire n'est pas nilpotent alors son ensemble limite est infini, voir K. Čulik II *et al* [19]. Un automate cellulaire nilpotent atteint son ensemble limite en un temps fini, *i.e.* il existe un temps t_0 tel que $\Lambda_F = F^{t_0}\left(S^{\mathbb{Z}^d}\right)$. La longueur de cette transitoire n'est cependant pas bornée récursivement en l'automate cellulaire :

Théorème 5 (K. Čulik II *et al* [20] et J. Kari [46])**.** *La nilpotence des automates cellulaires est indécidable.*

D'autres types d'attracteurs ont été proposés pour caractériser plus finement certains types de comportement. Ainsi, la notion d'ensemble limite a été restreinte aux configurations finies ou périodiques, voir K. Čulik II *et al* [19]. Ou encore, l'ensemble ultime $\omega_F(c)$ d'une configuration c est l'ensemble des points auxquels l'orbite de c adhère infiniment souvent, *i.e.* $\omega_F(c) = \bigcap_{t\in\mathbb{N}} \overline{\mathcal{O}_F(F^t(c))}$. L'*ensemble ultime* ω_F d'un automate cellulaire de fonction globale F est obtenu comme union des ensembles ultimes de ses configurations $\omega_F = \bigcup_{c\in S^{\mathbb{Z}^d}} \omega_F(c)$. L'ensemble ultime est stable par F, ce qui implique $\omega_F \subseteq \Lambda_F$. Pour en savoir plus, voir J. Cervelle [13] et P. Guillon et G. Richard [38].

Les premiers efforts de classification des automates cellulaires par leur dynamique visent, essentiellement, d'une part à tenter de formaliser les classifications expérimentales à la S. Wolfram [94], voir par exemple K. Čulik II et S. Yu [21], et d'autre part à formaliser la notion d'automate cellulaire chaotique, voir E. Formenti [33]. On notera en particulier la classification de P. Kůrka [52] qui classifie les automates cellulaires de dimension 1 en 4 classes selon leur sensibilité aux conditions initiales.

1.2 Groupages

Notre premier domaine de contribution à l'étude des automates cellulaires concerne la problématique de classification. Les classifications par groupage se distinguent des autres classifications établies par deux aspects. D'une part, alors que la plupart des classifications ont une approche topologique, visant à identifier finement le caractère chaotique des automates cellulaires classifiés, le groupage est issu d'une approche algébrique, visant à prendre en compte les phénomènes algorithmiques dans la dynamique des automates cellulaires. Un automate cellulaire en simule un autre si tous les diagrammes espace-temps du second sont des diagrammes espace-temps du premier, à une transformation régulière près de part et d'autre — les transformations respectant les symétries des automates cellulaires. D'autre part, le groupage ne décompose pas les automates cellulaires en un nombre fini de classes étiquetant un nombre fixé de propriétés — en caricaturant : les

automates très simples, les chaotiques et puis les autres — mais les répartit en la famille infinie dénombrable des classes d'équivalence d'une relation de pré-ordre. Notre apport a consisté, d'une part, en le couplage de la classification originelle purement algébrique de J. Mazoyer et I. Rapaport [65] avec la notion d'universalité intrinsèque pour définir le groupage dans son acceptation moderne et, d'autre part, à montrer que, dans cette extension du groupage, de nombreuses propriétés très étudiées des automates cellulaires, comme les propriétés immédiates, se caractérisent par des classes ou des idéaux de la structure : la compréhension de la structure de l'ordre et la compréhension de la dynamique des automates cellulaires se nourrissent mutuellement. Cette approche a ensuite été enrichie par les travaux de G. Theyssier [88]. Les versions courantes de deux articles de synthèse sur les groupages co-écrits avec M. Delorme, J. Mazoyer et G. Theyssier [U1, U2] sont disponibles en annexes A.1 et A.2.

1.2.1 Genèse

La classification par *groupage carré* de J. Mazoyer et I. Rapaport [65] considère les automates cellulaires de dimension 1 et de voisinage $\{-1, 0, 1\}$. Elle repose sur une relation élémentaire de pré-ordre algébrique. Un automate cellulaire (S, f) est un *sous-automate* d'un automate cellulaire (S', f'), noté $(S, f) \subseteq (S, f')$, si la table de transition du premier automate est une sous-table de la table de transition du second automate, *i.e.* s'il existe une application injective $\varphi : S \to S'$ telle que, pour tout triplet d'états $(x, y, z) \in S^3$, on a $f'(\varphi(x), \varphi(y), \varphi(z)) = \varphi(f(x, y, z))$. Cette relation est monotone en le nombre d'états. Aussi, J. Mazoyer et I. Rapaport la couplent à une transformation sur les tables de transition correspondant au changement d'échelle rencontré fréquemment dans les constructions algorithmiques à base de signaux (ainsi, la dernière étape de la construction de reconnaissance de nombre premier de P. C. Fischer [32] est un groupage des cellules par 3). L'itérée n fois f^n de la règle de transition f est définie par récurrence par $f^1 = f$ et

$$f^{n+1}(x_1, \ldots, x_{2n+3}) = f(f^n(x_1, \ldots, x_{2n+1}), f^n(x_2, \ldots, x_{2n+2}), f^n(x_3, \ldots, x_{2n+3})) \quad .$$

La *puissance n*ième d'un automate cellulaire (S, f), notée $(S, f)^n$, est l'automate cellulaire $(S^n, f^n_\square)$ où $f^n_\square$ est l'action de f^n sur les blocs de n états :

$$f^n_\square\big((x_1,\ldots,x_n), (x_{n+1},\ldots,x_{2n}), (x_{2n+1},\ldots,x_{3n})\big) = \big(f^n{}_{(x_1,\ldots,x_{2n+1})}, \ldots, f^n{}_{(x_n,\ldots,x_{3n})}\big) \quad .$$

La relation de groupage carré se définit alors ainsi : un automate cellulaire (S, f) *est simulé* par un automate cellulaire (S', f'), noté $(S, f) \leqslant_\square (S', f')$, s'il existe deux puissances $m, n \in \mathbb{N}$ telles que $(S, f)^m \subseteq (S', f')^n$.

Théorème 6 (J. Mazoyer et I. Rapaport [65])**.** *La relation de simulation $\leqslant_\square$ est une relation de pré-ordre.*

Du caractère régulier et algébrique du groupage carré, J. Mazoyer et I. Rapaport [65] extraient des informations structurelles sur l'ordre induit : l'ordre possède une classe minimale qui consiste en l'automate trivial à état unique ; les automates cellulaires nilpotents forment une classe d'équivalence de niveau 1, c'est-à-dire sans classe intermédiaire entre elle et la classe minimale ; l'automate cellulaire identité et les translations à gauche et à droite sont dans 3 classes d'équivalence de niveau 1 distinctes. La structure de groupe des automates cellulaire $(\mathbb{Z}_p, +)$ avec p premier permet de montrer qu'ils sont chacun dans

une classe d'équivalente distincte de niveau 1. Enfin, ils obtiennent un résultat structurel qui montre la richesse de l'ordre :

Théorème 7 (J. Mazoyer et I. Rapaport [65]). *La relation de simulation $\leqslant_{\square}$ possède des chaînes infinies croissantes non bornées.*

Si le caractère algébrique montre toute sa puissance pour démontrer la séparation des propriétés des automates cellulaires en des classes distinctes et pour identifier certains comportements qualifiés de simples avec les premiers niveaux de l'ordre, l'absence d'élément maximal dans l'ordre n'est pas satisfaisante. En effet, il existe des automates cellulaires intrinsèquement universels, c'est-à-dire capables de simuler tout automate cellulaire (cette notion est discutée en détail dans la section 1.3) et il serait raisonnable qu'une classification définie par simulation admette une classe maximale correspondant à cette universalité.

Calcul, pré-ordres et universalité : le mélange de ces trois notions n'est pas inconnu en calculabilité. Ainsi, E. L. Post s'est intéressé à ce mélange en plusieurs occasions. En 1941, étudiant le pouvoir d'expression relatif des familles de fonctions booléennes, *i.e.* de fonctions $f : \{0,1\}^n \to \{0,1\}$, pour le calcul par circuit booléen, il munit les ensembles de fonctions booléennes d'une relation de simulation et étudie l'ordre induit, dont la classe maximale est la classe des familles booléennes universelles. Une famille $\{f_1, \ldots\}$ de fonctions booléennes est simulée par une famille $\{g_1, \ldots\}$ si toute fonction f_k s'exprime par compositions des g_i (l'usage explicite de constantes est interdit). E. L. Post [76] caractérise complètement l'ordre et associe à chacune de ses classes d'équivalence dénombrables un représentant fini canonique de cette classification décidable. En 1944, il introduit la notion de réduction en calculabilité — un pré-ordre sur les ensembles récursivement énumérables — avec les réductions *many-one*, *one-one* et *Turing*, caractérise la classe maximale des deux premières réductions comme l'ensemble des ensembles créatifs, montre l'existence de classes intermédiaires pour ces réductions, les ensembles simples, et pose le célèbre problème de Post, voir E. L. Post [77] et P. G. Odifreddi [75].

1.2.2 Extension du groupage

Notre extension du groupage [T1, U1] repose sur une dissection du groupage carré pour en identifier les éléments fondamentaux, les étendre et les recomposer en un groupage plus général qui admet l'universalité comme classe maximale.

Relation sous-automate

Parmi les relations de pré-ordre élémentaire possibles, trois se distinguent [U2] : la relation sous-automate du groupage carré, la relation quotient (remplacer l'injectivité de φ par la surjectivité et inverser le sens de la relation) et la relation mixte (produit des deux relations précédentes). Le caractère algébrique de la relation sous-automate est renforcé par le résultat suivant, où les fonctions autarciques sont par définition les fonctions globales des automates cellulaires de voisinage $\{0\}$:

Théorème 8 ([T1, U1]). *Les fonctions globales des automates cellulaires sont les fonctions obtenues par clôture des fonctions autarciques et des translations élémentaires par composition, produit cartésien et sous-automate.*

La relation sous-automate possède aussi l'atout de s'exprimer simplement sur les diagrammes espace-temps : un automate cellulaire est sous-automate d'un second si les diagrammes espace-temps du premier sont (à un isomorphisme d'états près) des diagrammes espace-temps du second.

Fonctions globales

En tirant parti de la liberté de manipulation directe des fonctions globales des automates cellulaires qu'offre le théorème de Curtis-Hedlund-Lyndon, l'opération de passage à la puissance s'exprime algébriquement sur les fonctions globales, ce qui élimine le besoin de considérer uniquement certains voisinages. On définit le groupage carré pour tout automate cellulaire, quelque soit la dimension et le voisinage. Un automate cellulaire $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ est simulé par un automate cellulaire $G : S'^{\mathbb{Z}^d} \to S'^{\mathbb{Z}^d}$ s'il existe $m, n \in \mathbb{N}$ et une application injective $\varphi : S \to S'$ tels que :

$$\overline{\varphi} \circ b_S^m \circ F^m \circ b_S^{-m} = b_{S'}^n \circ G^n \circ b_{S'}^{-n} \circ \overline{\varphi}$$

où $\overline{\varphi}$ est l'extension canonique de φ, $b_S^n : S^{\mathbb{Z}^d} \to \left(S^{n^d}\right)^{\mathbb{Z}^d}$ est l'opérateur bijectif continu de construction de blocs et b_S^{-n} est l'inverse de b_S^n.

Une fois sous cette forme, les automates cellulaires commutant avec les translations, on identifie trois transformations élémentaires : regroupage spatial, découpage temporel et translation. Le *groupage rectangle* [T1, U1] se définit alors comme suit. Un automate cellulaire $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ est *simulé* par un automate cellulaire $G : S'^{\mathbb{Z}^d} \to S'^{\mathbb{Z}^d}$, noté $F \preccurlyeq G$, s'il existe $\langle m, n, k\rangle, \langle m', n', k'\rangle \in \mathbb{N}^d \times \mathbb{N} \times \mathbb{Z}^d$ et une application injective $\varphi : S \to S'$ tels que :

$$\overline{\varphi} \circ \sigma_k \circ B_S^m \circ F^n \circ B_S^{-m} = \sigma_{k'} \circ B_{S'}^{m'} \circ G^{n'} \circ B_{S'}^{-m'} \circ \overline{\varphi}$$

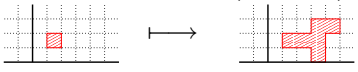
où σ_k est la translation de vecteur k et $B_S^m : S^{\mathbb{Z}^d} \to (S^m)^{\mathbb{Z}^d}$ est l'opérateur bijectif continu de construction de blocs et B_S^{-m} est l'inverse de b_S^m.

Le groupage rectangle possède une classe maximale car il prend en compte les simulations de l'universalité intrinsèque, voir section 1.3.

Transformations géométriques

Si la relation sous-automate s'exprime simplement sur les diagrammes espace-temps, les transformations de regroupage spatial, découpage temporel et translation aussi. Plus généralement, une transformation géométrique associe à tout diagramme espace-temps un nouveau diagramme espace-temps par découpage et recollage des cellules qu'il contient, le découpage ne dépendant pas du contenu. Plus formellement, une transformation géométrique est une paire $(k, \mathbf{\Lambda})$ où $k \in \mathbb{N}$ et $\mathbf{\Lambda}$ est le découpage

$$\mathbf{\Lambda} : \quad \mathbb{N} \times \mathbb{Z}^d \longrightarrow \left(\mathbb{N} \times \mathbb{Z}^d\right)^k \quad .$$



Les transformations du groupage rectangle sont quasiment les transformations **PCS**, qui sont la composition de transformations de regroupage spatial, de découpage temporel et de

translation régulière de la grille. L'unique différence réside dans le fait que les transformations **PCS** de regroupage spatial autorisent à découper l'espace par n'importe quelle pièce finie, non nécessairement connexe, qui pave l'espace, alors que le groupage rectangle se restreint aux pavés. En définissant les jolies transformations comme les transformations qui d'une part transforment l'ensemble des diagrammes espace-temps d'un automate cellulaire en exactement l'ensemble des diagrammes espace-temps d'un autre automate cellulaire et d'autre part ne sont pas dégénérées, c'est-à-dire respectent un minimum les dépendances de temporalité, nous avons montré le théorème suivant en exploitant la régularité des automates cellulaires :

Théorème 9 ([T1, U1])**.** *Les jolies transformations sont les transformations* **PCS**.

Les transformations utilisées dans notre extension du groupage sont donc les plus générales qui préservent le caractère géométrique de transformation des diagrammes espace-temps.

Groupage abstrait

Nous avons axiomatisé le groupage $\mathbf{\Phi_b}$ sous la forme d'une théorie de signature

$$\Big(\mathsf{Obj}, \mathsf{Trans};\ \mathbf{apply} : \mathsf{Obj} \times \mathsf{Trans} \to \mathsf{Obj},\ \mathbf{divide} \subseteq \mathsf{Obj} \times \mathsf{Obj},\ \mathbf{combine} : \mathsf{Trans} \times \mathsf{Trans} \to \mathsf{Trans}\Big)$$

où Obj est un ensemble d'objets, les automates cellulaires, Trans un ensemble de transformations, les transformations **PCS**, **apply** est l'opérateur de transformation, **divide** est la relation de simulation élémentaire et **combine** son opération de composition. La signature est munie des axiomes suivants, où les lettres latines (x, y, ...) sont les éléments de la sorte Obj, les lettres grecques (α, β, ...) et 1 sont les éléments de la sorte Trans, x^α dénote $\mathbf{apply}(x, \alpha)$, $x \mid y$ dénote $\mathbf{divide}(x, y)$ et $\alpha \cdot \beta$ dénote $\mathbf{combine}(\alpha, \beta)$:

$$\begin{aligned}
(B_1)\quad & \exists 1 \forall \alpha\, (\alpha \cdot 1 = \alpha \wedge 1 \cdot \alpha = \alpha) \ \wedge\ \forall \alpha \forall \beta \forall \gamma\, ((\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)) \\
(B_2)\quad & \forall x\, (x^1 = x) \ \wedge\ \forall x \forall \alpha \forall \beta \left((x^\alpha)^\beta = x^{\alpha \cdot \beta}\right) \\
(B_3)\quad & \forall x\, (x \mid x) \ \wedge\ \forall x \forall y \forall z\, ((x \mid y \wedge y \mid z) \to x \mid z) \\
(B_4)\quad & \forall x \forall y \forall \alpha\, (x \mid y \to x^\alpha \mid y^\alpha) \\
(B_5)\quad & \forall \alpha \forall x \exists y\, (x \mid y^\alpha) \\
(B_6)\quad & \forall \beta \exists \gamma \forall \alpha \exists \delta\, (\alpha \cdot \gamma = \beta \cdot \delta)
\end{aligned}$$

Dans ce cadre, la relation de simulation $x \leqslant y$ se définit syntaxiquement pour tout $x, y \in \mathsf{Obj}$ par la formule $\exists \alpha \exists \beta \left(x^\alpha \mid y^\beta\right)$. Nous avons montré [T1, U1] que cette relation est un pré-ordre. Nous obtenons ainsi une notion de groupage abstrait.

Cette axiomatisation permet d'analyser finement le problème issu des transformations **PCS** générales. Dans le cas du groupage carré, les transformations commutent : la composition d'une puissance $n$ième et d'une puissance $m$ième est une puissance mn. Ce n'est plus le cas dans le cadre du groupage rectangle à cause des translations, et ce n'est pas du tout le cas pour les transformations **PCS** car la composition des figures pavant l'espace n'est pas commutative. L'axiome (B_6) remplace la commutativité par une version forte de la propriété du diamant. La restriction du groupage aux pavés est due à notre ignorance d'une plus grande famille admissible de regroupages spatiaux.

Problème ouvert 2. *Caractériser les sous-monoïdes pour la composition des pièces finies pavant l'espace qui admettent la propriété du diamant.*

En se restreignant aux pavés, le groupage rectangle satisfait les axiomes et nous atteignons l'objectif initialement fixé :

Théorème 10 ([T1, U1])**.** *La relation de simulation $\preccurlyeq$ est une relation de pré-ordre dont les automates cellulaires intrinsèquement universels forment la classe d'équivalence maximale.*

1.2.3 Structure

Une fois défini un nouveau pré-ordre dans lequel l'intrinsèque universalité joue le rôle de maximum, nous avons montré que la structure du nouvel ordre induit est tout aussi riche et qu'il est possible d'obtenir des résultats comparables à ceux connus pour le groupage carré, le haut de l'ordre et des idéaux en plus.

Structure générale

Tout comme dans le cas du groupage carré, l'ordre induit du groupage rectangle possède une structure complexe. Si le produit cartésien est compatible avec l'ordre, au sens où il définit un majorant, ce n'est pas une borne supérieure. De fait :

Théorème 11 ([T1, U2])**.** *L'ordre induit par le groupage n'est ni un sup-demi-treillis ni un inf-demi-treillis.*

On y trouve aussi des chaînes infinies :

Théorème 12 ([T1, U2])**.** *Le groupage admet des chaînes infinies croissantes et décroissantes.*

D'une manière générale, la structure de l'ordre est encore peu comprise, même si on y plonge toute sorte de sous-ordres étranges, voir [U2].

Bas de l'ordre

Malgré l'ajout de transformations plus complexes, le bas de l'ordre ressemble beaucoup à celui du groupage carré. Les classes identifiées en bas de l'ordre correspondent bien à des comportements dynamiques simples. Ainsi, la classe minimale est constituée de l'unique automate cellulaire trivial à état unique. Au niveau 1 de l'ordre, on trouve : la classe contenant exactement les automates cellulaires nilpotents, la classe contenant exactement les compositions de translations et d'automates cellulaires périodiques, une classe distincte pour chaque $(\mathbb{Z}_p, +)$ avec p premier. Cette liste n'est pas exhaustive.

Le produit cartésien peut être utilisé pour figer des translations relatives, ainsi les produits cartésiens de n translations forment une structure de treillis dépendant des rapports de co-linéarité entre les translations.

Haut de l'ordre

Le haut de l'ordre est constitué de la classe maximale des automates cellulaires intrinsèquement universels. En un certain sens, il s'agit des automates cellulaires compliqués les plus simples à construire : c'est la somme des comportements de tous les autres. À partir de l'indécidabilité de l'appartenance à cette classe, on obtient quelques informations sur le haut de l'ordre. Ainsi, il n'existe pas de famille finie de classes d'équivalence formant exactement un niveau juste en dessous de la classe maximale.

Idéaux et filtres

L'absence de structure de demi-treillis n'empêche pas de définir une notion faible d'idéal et de filtre. Un idéal est un ensemble d'éléments stable vers le bas, *i.e.* pour tout automate cellulaire F appartenant à l'idéal et pour tout automate cellulaire G, si $G \preccurlyeq F$ alors G est dans l'idéal. Un filtre est un ensemble d'éléments stable vers le haut.

Les propriétés immédiates et certaines propriétés dynamiques forment naturellement des filtres et des idéaux, voir [T1, U2]. Par exemple :

Théorème 13 ([T1, U2]). *L'ensemble des automates cellulaires injectifs (resp. surjectifs) forme un idéal principal (resp. non principal en dimension supérieure à 2).*

Plus généralement, le groupage est compatible avec des notions algorithmiques de complexité. Ainsi, la complexité de vérification d'un motif est compatible avec le groupage (la complexité évolue de constante pour l'automate trivial à P-complet pour les automates intrinsèquement universels). De même, l'ordre est compatible avec les réductions many-one pour le problème de décision d'apparition d'un motif (de récursif pour l'automate trivial à $\mathbf{0}'_m$-complet, *i.e.* créatif, pour les automates intrinsèquement universels).

1.2.4 Poursuite des travaux

Si nous avons intégré l'universalité au groupage, cette classification, qui tend à caractériser la complexité algébrique, voire algorithmique, n'en est encore qu'à ses balbutiements. En dehors des problèmes plus techniques de compréhension de l'ordre dont on trouvera une discusion dans [T1, U1, U2], trois pistes de développement de ces travaux nous semblent pertinentes.

Une première piste concerne le choix de la relation élémentaire. Les groupages $\preccurlyeq_s$ et $\preccurlyeq_m$ sont obtenus à partir des relations quotient et mixte. Leur relation au groupage rectangle est la suivante :

Théorème 14 ([U2, 88]). *$\preccurlyeq$ et $\preccurlyeq_s$ sont des sous-relations incomparables de $\preccurlyeq_m$.*

Faut-il considérer la relation mixte plutôt que la relation sous-automate ? Qu'y gagne-t-on ? Qu'y perd-t-on ? Les deux problèmes suivants pourraient aider à répondre à ces questions :

Problème ouvert 3. *Les classes universelles de $\preccurlyeq$ et $\preccurlyeq_m$ coïncident-elles ?*

Problème ouvert 4. *Existe-t-il une classe maximale pour $\preccurlyeq_s$?*

Une deuxième piste est motivée par la problématique des systèmes complexes. En bas de l'ordre se trouvent les propriétés dynamiques simples. En haut de l'ordre on retrouve la complexité la plus simple : l'ensemble des comportements possibles. La véritable complexité est *au milieu* de l'ordre. En effet, pour tout entier n, il existe une classe d'équivalence pour laquelle le plus petit automate cellulaire de la classe est de complexité au moins n (pour toute notion de complexité raisonnable comme le nombre d'états à voisinage fixé ou le produit taille du voisinage par nombre d'états). Qui sont ces automates cellulaires ? Quelle est leur dynamique ?

Problème ouvert 5. *Construire une famille d'automates cellulaires de complexité, en nombre d'états minimal dans leur classe d'équivalence pour un voisinage fixé, arbitrairement grande pour le groupage.*

Une troisième piste concerne la régularité de l'ordre. Le groupage souffre de deux limitations. D'une part, l'absence de structure régulière de treillis, qui est due à l'irréversibilité des transformations regroupant les états. Faut-il autoriser le dépaquetage d'états ? D'autre part, l'analogie avec la calculabilité, qui reste encore faible. Dans un modèle de calcul classique, deux types d'objets sont présents : les indices de fonctions et les fonctions elles-mêmes. En définissant le groupage sur des attracteurs bien choisis (ensemble limite ou ensemble ultime), on peut espérer obtenir une classification plus calculatoire et trouver des pistes pour développer une axiomatique qui se rapproche des systèmes acceptables de programmation dans le style de H. Rogers Jr. [85].

Problème ouvert 6. *Définir une calculabilité des automates cellulaires où le groupage joue le rôle des réductions, où la notion de fonction récursive est remplacée par une notion de dynamique globale.*

1.3 Universalités

Notre second domaine de contribution à l'étude des automates cellulaires concerne l'universalité et, plus précisément, l'universalité intrinsèque. L'universalité est une propriété essentielle des modèles de calcul. Avec l'application partielle, le célèbre *smn*, elle constitue l'un des deux ingrédients des systèmes acceptables de programmation, voir [75, 85]. L'universalité met en avant la capacité d'un système à se décrire en interne, à manipuler syntaxiquement une représentation de lui-même. Aussi, il n'est pas étonnant que l'universalité soit l'une des notions les plus anciennes dans l'étude des automates cellulaires. Cependant, la notion classique d'universalité, obtenue en simulant le comportement de machines séquentielles, si elle est une notion algorithmique importante, souffre de plusieurs inconvénients pour l'étude des automates cellulaires en tant que systèmes complexes. Tout d'abord, c'est une notion extrinsèque, qui fait intervenir un objet d'une autre nature, le modèle de calcul séquentiel simulé. Ensuite, la propriété d'être universel est difficile à formaliser. Enfin, il existe des automates cellulaires universels à la dynamique très pauvre qui calculent très lentement. De fait, il existe une meilleure notion d'universalité : l'universalité intrinsèque. Un automate cellulaire est intrinsèquement universel s'il simule, en un sens raisonnable, tout autre automate cellulaire. Cette classe d'automates cellulaires est caractérisée, comme nous allons le voir, par la classe maximale du groupage. La distinction des différentes formes d'universalité et une tentative de formalisation ont été proposées

par B. Durand et Zs. Róka [29]. Notre contribution a consisté en la poursuite de cette distinction et en une étude fine de l'universalité intrinsèque : formalisation, caractérisation, décidabilité, exhibition de petits automates appartenant à la classe. Alors que nos deux autres domaines de contribution à l'étude des automates cellulaires sont plus prospectifs, l'étude de l'universalité, dont les contours sont bien délimités, peut être considérée comme quasiment close. Le lecteur trouvera en annexe A.3 un état de l'art sur l'universalité dans les automates cellulaires [C5, L1].

1.3.1 Genèse

Élément essentiel de la calculabilité, l'universalité apparaît dans l'article fondateur d'A. M. Turing [89] où une machine de Turing universelle est construite explicitement. Dans une approche plus axiomatique, une énumération acceptable [75] des fonctions récursives partielles $\varphi_n : \mathbb{N} \rightharpoonup \mathbb{N}$ est munie d'un crochet bijectif récursif $\langle\rangle : \mathbb{N}^2 \to \mathbb{N}$ pour lequel il existe un indice u vérifiant, pour toute paire d'entiers $m, n \in \mathbb{N}$:

$$\varphi_u(\langle m, n\rangle) \simeq \varphi_m(n)$$

où $x \simeq y$ si ou bien x et y sont tous deux non définis ou bien $x = y$. On notera qu'il n'est pas défini de propriété d'universalité mais une fonction universelle spécifique. La situation des fonctions est très différente de celle des problèmes de décision. En effet, pour ces derniers, la notion d'ensemble créatif, $\mathbf{0}'_m$-complet, caractérise bien la propriété d'universalité. Dans le cas des fonctions et des machines, malgré une proposition de M. D. Davis [22] reposant sur les ensembles créatifs, il n'y a pas de consensus autour d'une définition satisfaisante.

L'absence de définition formelle n'a pas empêché la recherche et la fabrication pour divers modèles de calcul de petites machines universelles, *pour une notion raisonnable d'universalité*. M. Minsky [68] introduit deux outils fondamentaux pour la construction de petites machines universelles : les machines à compteurs et les systèmes de Post. Grâce à ces modèles de calcul très minimaux, des machines de Turing à peu d'états et de transitions ont été construites, voir D. Woods et T. Neary [96] pour un panorama.

Les premiers automates cellulaires universels sont construits en dimension 2 par la simulation de circuits booléens. Les composants élémentaires (fils, signaux, portes logiques, croisements, *fan-out*) sont construits, permettant le câblage de toute machine à état fini. Ces circuits permettent de coder en une configuration infinie mais ultimement périodique toute machine de Turing. Leur universalité pour le calcul est donc établie. Cependant, ces automates cellulaires possèdent aussi une forme d'universalité intrinsèque : la règle de transition de tout automate cellulaire peut être codée sous la forme d'un circuit booléen ; en disposant des copies de ce circuit en une grille régulière et en les reliant par des fils, l'automate universel contient dans ses diagrammes espace-temps les diagrammes espace-temps de tout automate cellulaire de même dimension. Cette forme d'universalité intrinsèque est identifiée par E. R. Banks [6] qui la généralise à la dimension 1 en transformant tout automate cellulaire intrinsèquement universel de dimension 2 en un automate cellulaire intrinsèquement universel de dimension 1 à grand voisinage.

Cependant, les travaux d'E. R. Banks passent relativement inaperçus et en dimension 1 la communauté se focalise sur l'universalité Turing, cherchant à construire des automates cellulaires universels les plus simples possibles afin d'exhiber des mécanismes de

calcul typiquement cellulaires. Au début des années 90, le plus petit automate cellulaire connu universel pour le calcul en dimension 1 avec voisinage $\{-1, 0, 1\}$ à 7 états est dû à K. Lindgren et M. G. Nordahl [58]. La notion d'universalité intrinsèque est redécouverte par J. Albert et K. Čulik II [1] qui construisent un automate cellulaire à 14 états.

1.3.2 Universalité pour le calcul

Si une définition formelle d'universalité dans le cadre séquentiel est difficile, elle devient extrêmement délicate pour les automates cellulaires. En effet, de nouveaux problèmes voient le jour : comment coder récursivement, sans tricher, la paire $\langle m, n\rangle$ en une configuration initiale potentiellement infinie ? sur quel critère décider de l'arrêt du calcul ? comment décoder récursivement la sortie à partir de la configuration finale infinie ? B. Durand et Zs. Róka [29] ont discuté la difficulté de cette définition, ses dangers, et ont proposé une définition formelle. Cependant, comme nous l'avons montré, cette définition, qui doit accepter toutes les constructions antérieures, ne résiste pas à des tentatives de codage pernicieuses qui montrent l'universalité selon cette définition d'automates cellulaires très simples.

L'étude du groupage nous a permis de montrer que les deux notions d'universalité sont distinctes : si tout automate cellulaire intrinsèquement universel est universel pour le calcul (il lui suffit de simuler un tel automate), la réciproque est fausse, ce qui réfute les affirmations informelles de S. Wolfram [95].

Théorème 15 ([T1, U2, 88])**.** *Il existe des automates cellulaires universels pour le calcul non intrinsèquement universels.*

Si l'universalité pour le calcul présente peu d'intérêt dans l'étude de la dynamique des automates cellulaires, M. Cook [17] a exhibé un automate cellulaire à 2 états, la règle 110, qui est universel pour le calcul. Bien que T. Neary et D. Woods [73] aient amélioré le résultat en montrant la P-complétude de sa vérification de motif, nous ne savons pas prouver à ce jour si cette règle est ou non intrinsèquement universelle.

1.3.3 Universalité intrinsèque

Nous avons proposé, après avoir étudié en détail la littérature, de définir l'universalité intrinsèque à travers la notion de forte simulation. Un automate cellulaire $F : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ est *fortement simulé* par un automate cellulaire $G : S'^{\mathbb{Z}^d} \to S'^{\mathbb{Z}^d}$ s'il existe $\langle m, n, k\rangle \in \mathbb{N}^d \times \mathbb{N} \times \mathbb{Z}^d$ et une application injective $\varphi : S \to S'$ tels que :

$$\overline{\varphi} \circ F = \sigma_k \circ B_{S'}^m \circ G^n \circ B_{S'}^{-m} \circ \overline{\varphi}$$

Tout groupage abstrait qui possède un objet fortement universel a sa classe maximale composée exactement des objets fortement universels :

Théorème 16 ([T1, U1])**.** *Les automates cellulaires intrinsèquement universels sont fortement intrinsèquement universels.*

Une fois cette notion formalisée et les liens avec le groupage établis, nous nous sommes intéressés à la décidabilité de cette propriété :

Théorème 17 ([T1, C3]). *L'intrinsèque universalité est indécidable.*

D'un point de vue algorithmique, programmer un automate cellulaire intrinsèquement universel c'est construire un automate qui effectue deux fonctions principales : déplacer l'information codant les états de l'automate simulé et calculer en appliquant la règle de transition locale sur ces données. Le caractère non borné du groupage carré montre que cette analogie algorithmique a un sens : il n'est pas possible d'effectuer en même temps le déplacement d'information et le calcul.

Théorème 18 ([T1, U1]). *Il n'existe pas d'automate cellulaire intrinsèquement universel en temps réel.*

Enfin, nous nous sommes intéressés à la construction d'automates cellulaires intrinsèquement universels les plus simples possibles pour mettre à jour des techniques de programmation sur automates cellulaires. Dans un premier temps, nous avons construit un automate cellulaire à 2 états et grand voisinage dont la règle est algébriquement simple, c'est une forme bilinéaire (une somme de produits des états de deux voisins distincts) :

Théorème 19 ([T1, C1]). *Il existe des automates cellulaires bilinéaires intrinsèquement universels à 2 états.*

Dans un second temps, nous avons construit un automate cellulaire intrinsèquement universel à très peu d'états en introduisant une nouvelle méthode de codage des fonctions booléennes.

Théorème 20 ([T1, C2]). *Il existe un automate cellulaire intrinsèquement universel à 6 états et voisinage* $\{-1, 0, 1\}$.

Plus récemment, G. Richard, à l'aide des techniques développées dans le cadre de sa thèse pour l'étude des particules et collisions — voir section 1.4 — a amélioré le record actuel du plus petit automate cellulaire intrinsèquement universel, dont on trouve une description détaillée dans [P2] :

Théorème 21 ([U3, P2]). *Il existe un automate cellulaire intrinsèquement universel à 4 états et voisinage* $\{-1, 0, 1\}$.

1.3.4 Poursuite des travaux

L'étude de l'universalité nous a permis de nourrir notre étude du groupage et des systèmes de particules et collisions, nous donnant à manipuler des automates cellulaires à la dynamique complexe. La poursuite des travaux sur l'universalité vise essentiellement à clore la chasse au plus petit automate cellulaire intrinsèquement universel. Pour cela, il convient tout d'abord de développer de nouveaux outils pour montrer la non universalité d'un automate cellulaire donné, en plus du groupage et des propriétés de complexité. Enfin, la question essentielle est la suivante.

Problème ouvert 7. *Existe-t-il un automate cellulaire intrinsèquement universel à 2 états et voisinage* $\{-1, 0, 1\}$ *?*

1.4 Particules et collisions

Notre troisième domaine de contribution à l'étude des automates cellulaires, plus récent, concerne les automates cellulaires à particules et collisions. Dans un modèle de calcul classique où les actions élémentaires sont locales, c'est-à-dire dans un modèle de machine, comme les machines de Turing, les données à traiter sont organisées spatialement et calculer consiste en deux actions élémentaires : d'une part déplacer les quanta d'information d'un point de l'espace à un autre, d'autre part combiner des quanta d'information localement proches. Dans l'algorithmique des automates cellulaires, qui sont massivement parallèles, c'est la façon la plus courante de calculer. Si on observe les diagrammes espace-temps construits par ces algorithmes, l'information est organisée sous forme de signaux, de particules qui se déplacent linéairement dans un espace homogène et qui se combinent lors de leur rencontre, de leur collision, générant une perturbation localisée qui engendre de nouveaux signaux. Or, dans une vision anthropomorphique, on observe cette organisation de l'information dans les classifications expérimentales des automates cellulaires, telle la classe 4 de S. Wolfram [94] : c'est la forme complexe de dynamique dans laquelle l'information s'organise (par opposition aux comportements complexes chaotiques) en des structures simples localisées. Ces phénomènes interpellent à la fois les physiciens et biologistes qui y voient une similitude avec des phénomènes naturels, mais aussi l'informaticien qui y voit les composants élémentaires du calcul. Notre contribution à l'étude des automates cellulaires à particules et collisions vise, d'une part, à comprendre les objets élémentaires que sont les particules et collisions et, d'autre part, à mettre au point des outils qui permettent une manipulation algorithmique aisée de ces objets. Cette recherche a été développée dans le cadre des travaux de thèse de doctorat de G. Richard [P2], thèse à laquelle le lecteur est renvoyé pour une discussion plus fournie. Nous avons joint en annexes A.4 et A.5 les articles [C7, U4] correspondant aux résultats techniques décrits ci-après.

1.4.1 Genèse

Le premier automate cellulaire construit par J. von Neumann [74], en dimension 2, partitionne son ensemble d'états en deux ensembles distincts : la plupart des états sont quiescents, ils restent inactifs et constants jusqu'à l'arrivée dans leur voisinages d'états signaux. Les signaux se déplacent de manière rectiligne, modifiant leur trajectoire ou leur état lors de leur rencontre avec certains états. Cette organisation de l'information, très naturelle, correspond, si on l'observe dans l'espace-temps, à des déplacements linéaires de particules (les signaux mais aussi les états spécifiques) qui engendrent des perturbations lors de leurs collisions. Cette notion de particule est encore plus explicite si on observe les *gliders* du *jeu de la vie* de J. H. Conway [10] utilisés pour démontrer l'universalité pour le calcul de cet automate cellulaire. L'utilisation de particules et de collisions n'est pas fortuite. Ainsi, dans la course aux plus petits automates cellulaires universels, l'utilisation de particules pour optimiser le nombre d'états est la règle. On en trouvera une discussion explicite dans la construction de K. Lindgren et M. G. Nordahl [58], mais c'est aussi l'outil utilisé dans [C2, U3]. Pour démontrer l'universalité de la règle 110, l'approche de M. Cook [17] consiste à étudier les particules de la règle et leurs collisions pour en extraire les mécanismes nécessaires à sa démonstration ; voir G. Richard [82] pour une preuve complète allégée de l'universalité de la règle 110 utilisant le formalisme des particules et

collisions décrit ici.

Les signaux, dont J. Mazoyer et V. Terrier [66] proposent une formalisation, sont un des outils essentiels de l'algorithmique cellulaire. En effet, ils permettent une algorithmique géométrique : on raisonne tout d'abord dans un modèle continu en discutant des pentes relatives de signaux à construire et de la combinaison de ces signaux lors des collisions pour effectuer un calcul, puis on discrétise les traits de construction sous la forme de signaux, voir par exemple P. C. Fischer [32] ou encore J. Mazoyer [62].

Ces outils du calcul ne sont pas, *a priori*, des objets d'étude de la dynamique des automates cellulaires. Pourtant, S. Wolfram [94] identifie comme forme de complexité organisée des automates cellulaires l'émergence de structures simples qui se déplacent linéairement et interagissent ponctuellement lors de leur rencontre. On observe par exemple ces structures dans le cas de la règle 54, voir figure 1.1. La figure 1.4 représente un diagramme espace-temps typique d'un tel automate cellulaire. Il s'agit de l'automate cellulaire à 3 états de règle de transition $f(x, y, z) = \lfloor 6430564760289/3^{9x+3y+z} \rfloor \pmod 3$. La configuration initiale a été obtenue à l'aide d'un générateur de nombres pseudo-aléatoires. Pourtant, une grande partie du diagramme est composée de motifs bipériodiques, des fonds, voir figure 1.5. À l'interface entre exactement deux de ces fonds, on trouve des particules, des motifs périodiques, voir figure 1.6. Enfin, à l'interface entre plusieurs fonds, lorsque des particules se rencontrent, on observe des collisions, voir figure 1.7, des perturbations bornées. Ces constatations soulèvent plusieurs questions : pourquoi ces objets apparaissent-ils dans la plupart des diagrammes espace-temps de ces automates cellulaires ? quelle est la richesse des interactions d'un ensemble de particules et de collisions données ?

L'étude des automates cellulaires à particules et collisions s'est développée dans les années 90, adoptant le point de vue d'une mécanique des particules, voir W. Hordijk *et al* [43]. Nous proposons une nouvelle approche de la combinatoire des particules et des collisions en considérant les diagrammes espace-temps de ces automates cellulaires comme des pavages plutôt que comme des séquences de configurations.

1.4.2 Système de particules et collisions discrètes

Notre première contribution concerne la caractérisation des objets particule et collision. Les diagrammes espace-temps des automates cellulaires de dimension 1 sont des mots planaires satisfaisant des contraintes locales : ce sont les éléments d'un sous-shift de type fini, c'est-à-dire un pavage du plan, voir le chapitre 2. Un *fond* est un pavage bipériodique. Une *particule* est un pavage qui admet un vecteur de périodicité et est ultimement périodique selon la direction orthogonale à ce vecteur. Une *collision* est un objet plus complexe. Soit $\lhd_v(u, u')$ la portion angulaire du plan à droite de u de pointe $v \in \mathbb{Z}^2$ et délimitée par les vecteurs $u, u' \in \mathbb{Z}^2$. Formellement,

Définition 2. Une *collision* est un pavage $\mathfrak{C}$ pour lequel il existe un entier k et une suite cyclique de n vecteurs $(u_i) \in (\mathbb{Z}^2)^{\mathbb{Z}_n}$ tels que, pour tout $i \in \mathbb{Z}_n$, pour tout z dans $\lhd_{ku_i}(u_{i-1}, u_{i+1})$ le pavage est u_i-périodique en z, *i.e.* $\mathfrak{C}(z) = \mathfrak{C}(z + u_i)$.

La figure 1.8 présente un exemple de collision formelle et la représentation graphique correspondante. L'objet collision, s'il correspond à une intuition simple de rencontre de

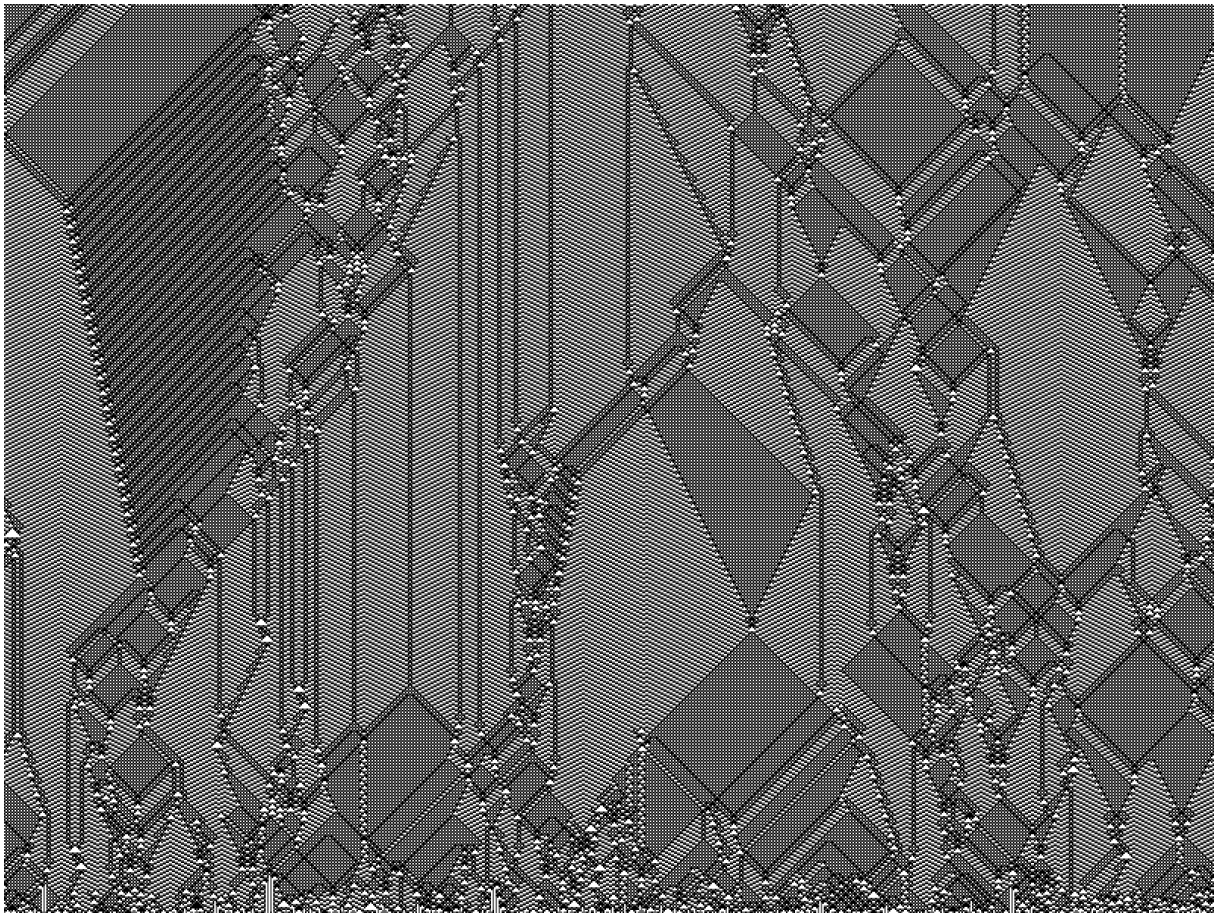


FIGURE 1.4 – diagramme espace-temps d'un automate cellulaire à particules et collisions

FIGURE 1.5 – des fonds

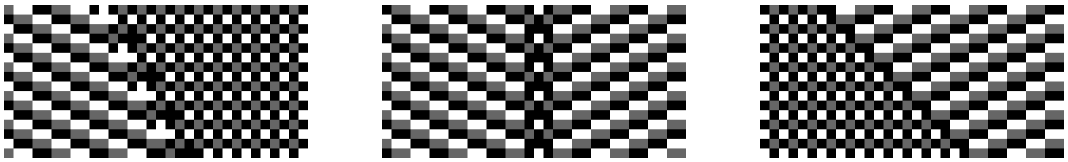


FIGURE 1.6 – des particules

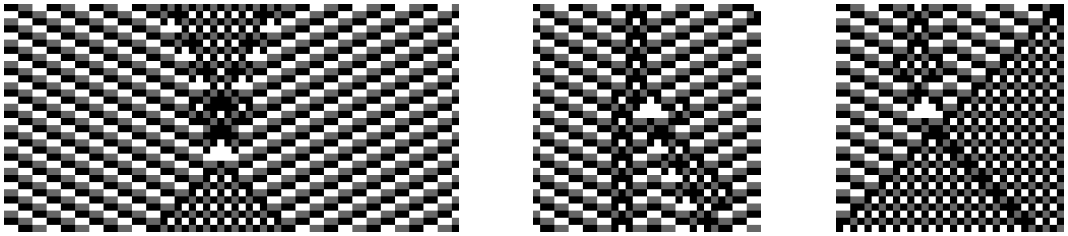
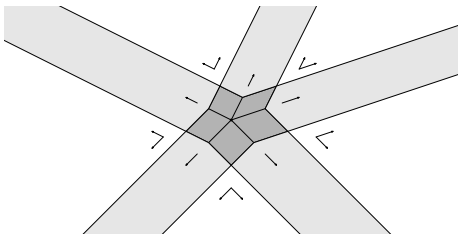


FIGURE 1.7 – des collisions

particules, est complexe à définir. Nous nous sommes intéressés, avec G. Richard, à expliquer l'apparition de cet objet comme caractérisation d'une classe de langages reconnus par des machines à états finis. Une $\mathbb{Z}^2$-k-carte est un pavage engendré par un automate



$k = 2$
$u_0 = \binom{1}{2}$
$u_1 = \binom{3}{1}$
$u_2 = \binom{2}{-2}$
$u_3 = \binom{-2}{-2}$
$u_4 = \binom{-2}{1}$

FIGURE 1.8 – Caractérisation des collisions

déterministe à k compteurs. Nous avons montré dans [U4] que les collisions correspondent exactement à la classe des pavages reconnus par les automates à 2 compteurs à boussole, c'est-à-dire mémorisant un point précis du plan :

Théorème 22 ([U4]). *Toute $\mathbb{Z}^2$-2-carte apériodique est une collision.*

Notre seconde contribution concerne l'algorithmique des particules et des collisions. Une opération naturelle sur les pavages est la construction de patchwork : connaissant la portée des contraintes locales, il est possible de découper puis de recoller des morceaux de pavages pour en former un nouveau si on prend soin de vérifier que les contraintes des morceaux voisins sont identiques dans les pavages d'origine. Ainsi, une opération de caténation naturelle apparaît sur les collisions : une famille finie de collisions peut être assemblée en un patchwork en reliant des particules identiques entre elles. Une telle caténation forme un plongement planaire dont les sommets sont étiquettés par des collisions et les arêtes par des particules. Les constructions algorithmiques sur automates cellulaires consistent généralement à construire d'abord un tel graphe puis à vérifier que les contraintes qu'il engendre sont satisfiables, c'est-à-dire qu'il est possible d'associer à chaque arête un nombre de répétition de son motif de sorte que le collage obtenu forme bien un patchwork. Cette vérification peut être automatisée :

Théorème 23 ([C7]). *L'ensemble des pondérations valides d'une caténation finie est un ensemble semi-linéaire récursif.*

Ce résultat a une réelle utilité en programmation. Ainsi, en étendant le résultat à des familles de caténations infinies régulières, G. Richard [P2] obtient des constructions d'universalité lisibles dont la vérification des contraintes est automatisable [U3, 82].

1.4.3 Poursuite des travaux

Une première problématique sur les automates à particules et collisions est de résoudre le mystère de l'émergence de ce type d'automate cellulaire dans les classifications expérimentales.

Problème ouvert 8. *Caractériser les automates cellulaires pour lesquels on observe expérimentalement des particules et des collisions.*

Une fois cette question résolue, se posera la question de l'existence d'un niveau suivant, voire d'une hiérarchie, d'auto-organisation : les automates à particules et collisions les plus complexes s'organisent-ils en meta-particules et meta-collisions ou une autre forme de complexité émerge-t-elle ?

Une seconde problématique concerne les aspects algorithmiques. Il s'agit d'unifier l'approche présentée ci-dessus avec les outils algorithmiques classiques sur automates cellulaires, afin de développer des langages et des outils qui permettent d'exprimer simplement les constructions géométriques et de démontrer leur validité.

Enfin, l'utilisation des automates cellulaires à particules et collisions pour la construction d'automates intrinsèquement universels incite à étudier de plus près les liens entre ces automates et les automates cellulaires conservatifs, voir la thèse de V. Bernardi [P1]. Les fonds et les particules sont extrêmement conservatifs, de par leur périodicité, alors les collisions sont le lieu des créations et dissipations d'énergie.

CHAPITRE

2 INDÉCIDABILITÉS, MACHINES ET APÉRIODICITÉS

« When the term "machine" is used in ordinary discourse, it tends to evoke an unattractive picture. It brings to mind a big, heavy, complicated object which is noisy, greasy, and metallic; perfoms jerky, repetitive, and monotonous motions; and has sharp edges that may hurt one if he does not maintain sufficient distance. »

Marvin L. Minsky, in *Computation : Finite and Infinite Machines* [68, p. 1]

2.1 Modèles de calcul et indécidabilité

Les résultats d'indécidabilité sont souvent perçus comme des résultats négatifs : ils indiquent l'impossibilité algorithmique de décider une propriété, de résoudre un problème. Cependant, lorsqu'ils sont obtenus par réduction, en plongeant l'ensemble des calculs possibles à l'intérieur de la propriété considérée, ils indiquent aussi une richesse de comportement. Notre démarche dans ce chapitre est d'exhiber des propriétés dynamiques de systèmes complexes qualifiées de simples, comme les classes 1 et 2 de la classification de S. Wolfram [94], et qui soient indécidables. Un tel résultat est positif dans le sens où il identifie des phénomènes calculatoires au coeur de la dynamique de systèmes complexes : le calcul est une source de complexité. Afin de plonger le calcul au coeur des propriétés des automates cellulaires, c'est-à-dire de programmer ces propriétés, nous rappelons dans la section 2.1.1 les rudiments de calculabilité nécessaires. Dans la section 2.1.2, nous présentons quelques résultats classiques d'indécidabilité et les méthodes pour les obtenir. Enfin, la section 2.1.3 est dédiée à l'introduction des problèmes des sections 2.2 et 2.3 et des liens étroits qu'ils entretiennent avec la notion d'apériodicité.

2.1.1 Calculabilité et machines

La théorie de la calculabilité, s'appuyant sur la thèse de Church-Turing, a pour objet l'étude des fonctions récursives et des problèmes de décision, de la frontière entre calculable et non calculable. Nous rappelons ici quelques éléments en adoptant le point de vue des modèles de calcul de type *machine* : une machine à état fini contrôlant un ou plusieurs organes mémoire sur lesquels elle exerce des actions élémentaires. L'indexation du temps par les transitions d'état de la machine et le caractère discret des organes permet de plonger facilement le calcul de la machine dans des objets discrets. Pour plus de détails concernant la calculabilité, en particulier pour une discussion sur les codages raisonnables évoqués ci-dessous, le lecteur est incité à consulter H. Rogers Jr. [86] et P. G. Odifreddi [75]. Pour un point de vue machine sur le calcul et une étude de différents modèles de calcul, le lecteur trouvera dans M. L. Minsky [68] les discussions et détails techniques nécessaires. Le point de vue adopté pour décrire les machines de Turing, inspiré de P. Kůrka [53], est celui de [C6] qui permet d'assurer une dualité entre machine et topologie, comme dans le cas des automates cellulaires, et de traiter aisément le calcul réversible de manière syntaxique.

Une machine de Turing est constituée d'une machine à états finis qui agit sur un ruban biinfini muni d'une tête de lecture/écriture. La machine peut déplacer le ruban sous la tête, lire et écrire sur la case pointée par la tête. Formellement, une *machine de Turing* M est un triplet (S, Σ, T) où S est l'ensemble fini d'états, Σ est l'alphabet fini du ruban et $T \subseteq (S \times \{\leftarrow, \rightarrow\} \times S) \cup (S \times \Sigma \times S \times \Sigma)$ est la table de transition de la machine. Une configuration $\mathfrak{c}$ de la machine est une paire (s, c) où $s \in S$ est l'état de la machine et $c \in \Sigma^{\mathbb{Z}}$ est le contenu du ruban, la tête étant située sur la case 0. La machine transforme une configuration $\mathfrak{c}$ en une configuration $\mathfrak{c}'$ en une transition, noté $\mathfrak{c} \vdash \mathfrak{c}'$, en appliquant une instruction $\iota \in T$. L'application d'une instruction de *mouvement* $(s, \delta, t) \in T$ transforme une configuration (s, c) en la configuration $(t, \sigma_\delta(c))$ où, pour tout $z \in \mathbb{Z}$, $\sigma_\leftarrow(c)(z) = c(z+1)$ et $\sigma_\rightarrow(c)(z) = c(z-1)$. L'application d'une instruction de *lecture/écriture* (s, a, t, b) transforme une configuration (s, c) telle que $c(0) = a$ en la configuration $(t, \mu_b(c))$ où $\mu_b(c)(0) = b$ et, pour tout $z \in \mathbb{Z}^*$, $\mu_b(c)(z) = c(z)$.

Une machine de Turing est *déterministe* si au plus une instruction est applicable pour toute configuration. À une machine de Turing déterministe on associe un système dynamique discret à fonction globale partielle $(S \times \Sigma^{\mathbb{Z}}, F)$ où $F : S \times \Sigma^{\mathbb{Z}} \rightharpoonup S \times \Sigma^{\mathbb{Z}}$ associe à une configuration $\mathfrak{c}$ l'unique configuration $\mathfrak{c}'$ telle que $\mathfrak{c} \vdash \mathfrak{c}'$. L'espace des configurations est muni de la topologie de Cantor : la topologie produit de la topologie discrète sur S par la topologie produit sur $\mathbb{Z}$ de la topologie discrète sur Σ, voir section 1.1.3 (il est nécessaire de faire bouger le ruban plutôt que la tête pour obtenir naturellement une topologie compacte). Nous verrons à la section 2.3 comment tirer parti de cette approche topologique des machines de Turing.

Dans le cadre classique, on distingue une lettre particulière de l'alphabet, le symbole blanc B, et on considère les orbites des machines de Turing à partir des configurations B-finies. Une fois fixé un codage raisonnable des entiers dans les configurations B-finies, on associe à chaque machine de Turing M une fonction partielle $\varphi_M : \mathbb{N} \rightharpoonup \mathbb{N}$: ainsi $\varphi_M(m) \simeq n$ si, partant de la configuration codant m, l'orbite de la machine atteint une configuration d'arrêt, sans successeur, qui code n. Les fonctions calculables par machine de Turing sont exactement les fonctions récursives partielles. Le premier pas en calculabilité consiste à se convaincre de l'existence de fonctions non calculables par l'utilisation du procédé diagonal de Cantor : soit directement en remarquant que les fonctions de $\mathbb{N} \to \mathbb{N}$ ne sont pas dénombrables, soit explicitement en exhibant un problème qui n'est pas récursif. Le problème de l'arrêt consiste, étant donné une machine de Turing et une configuration B-finie, à décider si l'orbite issue de cette configuration est infinie, noté $\uparrow$, ou si la machine s'arrête au bout d'un temps fini, noté $\downarrow$.

Théorème 24 (A. M. Turing [89])**.** *L'arrêt des machines de Turing est indécidable.*

Une fois un codage raisonnable des machines de Turing fixé, ce résultat identifie un ensemble d'entiers récursivement énumérable et non récursif $\mathcal{K} = \{n \in \mathbb{N} | \varphi_n(n) \downarrow\}$. On peut alors construire de nouveaux ensembles non récursifs à l'aide des réductions, formalisées par E. L. Post [77]. Un ensemble d'entiers A se réduit à un ensemble d'entiers B pour la réduction *many-one*, noté $A \leqslant_m B$, s'il existe une fonction récursive totale $f : \mathbb{N} \to \mathbb{N}$ telle que, pour tout $x \in \mathbb{N}$, $x \in A \Leftrightarrow f(x) \in B$. Ainsi, pour montrer qu'un problème est indécidable, on montre souvent qu'on peut réduire un ensemble non récursif à l'ensemble associé. Considérons à titre d'exemple le théorème de Rice :

Théorème 25 (H. G. Rice [81])**.** *Soit $\mathcal{C}$ un ensemble de fonctions récursives partielles. L'ensemble des machines de Turing qui calculent une fonction de $\mathcal{C}$ est soit trivial soit non récursif.*

Pour démontrer ce résultat par réduction de $\mathcal{K}$ à l'ensemble des codes de machines C correspondant à un ensemble $\mathcal{C}$ non trivial fixé contenant une machine a mais pas la fonction définie nulle part $\perp$, on associe récursivement à tout entier $n \in \mathbb{N}$ la machine de Turing de code $f(n)$ qui simule l'exécution de φ_n sur l'entrée n et, lorsque cette simulation s'arrête, exécute φ_a sur l'entrée donnée. Par construction, $n \in \mathcal{K} \Leftrightarrow f(n) \in C$.

2.1.2 Indécidabilités

La relation de réduction $\leqslant_m$ est une relation de pré-ordre dont les classes d'équivalence sont les *degrés many-one*. L'ordre induit admet une classe minimale de fonctions non triviales, l'ensemble $\mathbf{0}_m$ des fonctions récursives, et une classe maximale, l'ensemble $\mathbf{0}'_m$ des ensembles créatifs, qui contient $\mathcal{K}$. Ainsi, lorsqu'on démontre l'indécidabilité d'un problème par réduction de $\mathcal{K}$, on montre que ce problème est difficile pour les ensembles récursivement énumérables. Pour une étude de la structure de l'ordre, voir P. G. Odifreddi [75]. L'ensemble des réductions que nous considérons dans ce mémoire sont des réductions *many-one* d'éléments de $\mathbf{0}'_m$.

Si les techniques utilisées pour démontrer l'indécidabilité d'une propriété varient en fonction des objets considérés, le squelette reste souvent le même. Il s'agit de faire une ou plusieurs réductions successives depuis un problème indécidable connu jusqu'à la propriété souhaitée, de manière à transformer les objets successivement en objets plus proches du résultat souhaité, parfois en préservant des propriétés secondaires. La fonction de réduction, récursive, est un programme, l'activité de réduire est une forme de programmation plongeant un objet dans un autre, par opposition aux programmations externes de la section 1.1.1 et internes de la section 1.1.2.

Le plus ancien problème de nature non informatique dont l'indécidabilité a été démontrée est sans doute le problème du mot pour les semigroupes, ou problème d'accessibilité pour les systèmes de Thue. Étant donné un alphabet Σ, un ensemble fini de paires de mots $u_i \leftrightarrow v_i$ et deux mots u et v sur Σ, il s'agit de décider s'il est possible de transformer u en v par réécriture successive de sous-mots u_i en v_i et réciproquement. Un système semi-Thue est un système dans lequel on ne s'autorise à réécrire u_i en v_i (noté $u_i \to v_i$) mais pas v_i en u_i. L'indécidabilité de l'accessibilité pour les systèmes de Thue a été démontrée par E. L. Post [78] par réduction du problème de l'arrêt des machines de Turing au problème de l'accessibilité pour les systèmes semi-Thue, puis réduction à l'accessibilité pour les systèmes de Thue. Le problème, purement algébrique, du mot pour les groupes — étant donné une présentation de groupe et un mot, décider si ce mot est égal à l'identité du groupe — a ensuite été démontré indécidable par réduction du problème du mot pour les semigroupes, voir J. L. Britton [11].

Le problème du mot est un problème où interviennent plusieurs infinis dans les paramètres. Ainsi, le nombre de paires de mots est non borné, ainsi que la taille des mots. Pour affiner la compréhension d'un tel problème, on peut fixer un des paramètres et regarder si l'indécidabilité résiste. Ainsi, le problème d'accessibilité pour les sytèmes semi-Thue à

3 règles est indécidable, voir Y. Matiyasevich et G. Sénizergues [61] qui contient en outre un historique du problème.

Un autre problème standard indécidable, de nature plus combinatoire, est le problème de correspondance de Post. Étant donné un alphabet et un ensemble fini de paires de mots (u_i, v_i) sur Σ, il s'agit de décider s'il existe une suite finie d'indices $(i_1, \ldots, i_n)$ tels que $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$. M. L. Minsky [67] obtient l'indécidabilité de ce problème par réduction du problème de l'arrêt des machines de Turing au problème de l'arrêt des machines à deux compteurs, puis réduction au problème de correspondance de Post. En modifiant la réduction, et en réduisant le problème de l'accessibilité d'un système semi-Thue à n règles au problème de correspondance de Post à $n+4$ règles, V. Claus [14] obtient l'indécidabilité du problème de correspondance de Post à nombre fixé de paires. Le résultat de Y. Matiyasevich et G. Sénizergues [61] permet de montrer que le problème est indécidable à partir de 7 paires. Le problème est décidable pour 2 paires, la question reste ouverte pour 3 à 6 paires. Nous établirons dans la section 2.2.3 un résultat comparable pour la pavabilité du plan par des polyominos.

L'indécidabilité est aussi présente au coeur de problèmes purs de combinatoire des mots. La concaténation de langages est définie par $XY = \{uv | u \in X, v \in Y\}$. Deux langages X et Y commutent si $XY = YX$. Tout langage X possède un centraliseur $Z(X)$, le plus grand langage pour l'inclusion qui commute avec X. J. H. Conway pose dans [16] la question de savoir si le centraliseur d'un langage rationnel est toujours rationnel. M. Kunc [54] a démontré que certains langages finis ont des centraliseurs non récursifs. S'inspirant de ce résultat, nous avons montré avec E. Jeandel dans [J2], présenté en annexe B.4, comment réduire le problème de l'arrêt d'un système de Post au centraliseur d'un langage rationnel par des réductions successives.

2.1.3 Apériodicités

L'indécidabilité traitée dans les sections 2.2 et 2.3 entretien des liens étroits avec certaines formes d'apériodicité. Considérons le problème de l'arrêt des machines de Turing d'un point de vue dynamique. Partant d'une configuration, l'orbite de cette configuration peut soit être finie, soit être infinie. Dans ce dernier cas, elle peut être ultimement périodique, c'est-à-dire repasser deux fois par une même configuration, ou apériodique. Les orbites finies et ultimement périodiques étant récursivement énumérables, le problème ne peut être indécidable que s'il existe des orbites apériodiques. Une autre conséquence est que le problème suivant est indécidable : décider, étant donné une machine de Turing et une configuration dont l'orbite est infinie, si cette orbite est ultimement périodique.

Les deux problèmes étudiés sont issus d'une même problématique. Au début des années 60, cherchant à décider l'*Entscheidungsproblem* (voir [31]) sur la classe préfixe $\forall\exists\forall$, H. Wang et J. R. Büchi introduisent deux problèmes de décision qui permettraient de répondre à cette question. Le problème sera résolu par A. Kahr *et al* [44] en démontrant l'indécidabilité d'une version faible du *Domino Problem* : la pavabilité avec diagonale contrainte. Les deux problèmes, qui sont présentés formellement dans leurs section respectives, sont les suivants :

Domino Problem *"Assume we are given a finite set of square plates of the same size with edges colored, each in a different manner. Suppose further there are infinitely many*

copies of each plate (plate type). We are not permitted to rotate or reflect a plate. The question is to find an effective procedure by which we can decide, for each given finite set of plates, whether we can cover up the whole plane (or, equivalently, an infinite quadrant thereof) with copies of the plates subject to the restriction that adjoining edges must have the same color." H. Wang [90]

Immortality Problem *"(T_2) To find an effective method, which for every Turing-machine M decides whether or not, for all tapes I (finite and infinite) and all states B, M will eventually halt if started in state B on tape I"* J. R. Büchi [12]

Dans ces deux problèmes, qui sont indécidables, la difficulté pour réduire le problème de l'arrêt des machines de Turing consiste en l'absence d'origine où coder la configuration initiale de la machine de Turing. Pour les pavages, il s'agit de forcer une tuile particulière à apparaître dans le pavage. Pour l'immortalité, il s'agit de forcer le calcul à repasser par la configuration initiale dans toute orbite. L'apériodicité offre une piste pour y arriver. Ainsi, par compacité, les jeux de tuiles qui ne pavent pas le plan et ceux qui admettent un pavage périodique sont récursivement énumérables. L'indécidabilité est donc à rechercher dans les jeux de tuiles dont tous les pavages sont apériodiques. Pour l'immortalité, par compacité, les machines qui s'arrêtent sur toute configuration et celles qui ont une orbite ultimement périodique sont récursivement énumérables. L'indécidabilité est donc à rechercher dans les machines dont toutes les orbites infinies sont apériodiques. Ces liens entre les problèmes nous incitent à poser un problème :

Problème ouvert 9. *Établir formellement la dualité entre les problèmes de pavage du plan et de l'immortalité des machines de Turing.*

2.2 Pavages

Notre premier domaine de contribution à l'étude de la décidabilité de propriétés dynamiques des systèmes complexes concerne l'étude des pavages. Les jeux de tuiles et les problèmes de pavabilité sont un élément essentiel de l'étude des propriétés des automates cellulaires. Au début des années 90, J. Kari montre à l'aide de ces outils que certaines propriétés immédiates comme l'injectivité et la surjectivité, qui sont décidables en dimension 1, sont indécidables à partir de la dimension 2. Dans un second temps, il démontre que la propriété d'être nilpotent, qui caractérise la classe 1 de S. Wolfram, est indécidable en dimension 1 et en dérive une sorte de théorème de Rice pour les ensembles limites d'automates cellulaires. Les démonstrations d'indécidabilité de propriétés d'automates cellulaires qui ne procèdent pas par réduction directe du problème de l'arrêt des machines de Turing réduisaient toutes, jusque récemment (voir section 2.3), des problèmes de pavabilité. Ainsi, nous avons obtenu l'indécidabilité de l'intrinsèque universalité, voir [C3]. Plus récemment, notre apport à l'étude de la pavabilité se situe en amont de ses liens avec les automates cellulaires. Afin de mieux cerner l'origine de l'indécidabilité et son codage dans les automates cellulaires, nous nous sommes intéressés tout d'abord à l'indécidabilité de la pavabilité du plan par un jeu de tuiles de Wang. Ce résultat obtenu en 1966 par R. Berger [8, 9] est très technique. Nous nous sommes efforcés à en trouver une démonstration plus simple en suivant les techniques classiques de preuve, voir [C4] présenté en annexe B.1. Dans le même temps, de nouvelles méthodes de preuve ont été

développées par B. Durand *et al* [30] et J. Kari [51]. Notre seconde contribution à l'étude des pavages concerne la pavabilité du plan par des polyominos. Une réduction immédiate depuis la pavabilité du plan par des tuiles de Wang en assure l'indécidabilité, mais nous avons obtenu un résultat plus fin en démontrant l'indécidabilité de la pavabilité du plan par un nombre borné polyominos, voir [U6] présenté en annexe B.2.

2.2.1 Pavabilité par des tuiles de Wang

Un jeu de tuiles de Wang est la donnée d'un ensemble fini de carrés unité aux côtés colorés qu'on agence en les disposant sans rotation ni réflexion sur la grille discrète $\mathbb{Z}^2$ de sorte à ce que les couleurs de part et d'autre d'une arête soient identiques. Plus formellement, une *relation domino* $\mathcal{R} \subseteq T \times T$ sur un alphabet fini T est une relation qui satisfait la propriété du domino :

$$\forall a, b, c, d \in T^4, \quad a\mathcal{R}c \wedge a\mathcal{R}d \wedge b\mathcal{R}d \rightarrow b\mathcal{R}c \quad .$$

L'ensemble de couleurs associées à $\mathcal{R}$ est l'ensemble des classes d'équivalence de la relation d'équivalence $\sim_{\mathcal{R}}$ définie par : $(a, c) \sim_{\mathcal{R}} (b, d)$ si $a\mathcal{R}c \wedge a\mathcal{R}d \wedge b\mathcal{R}d$. La couleur droite d'un élément $a \in T$ est la couleur $[a\rangle$ qui contient (a, b) où $b \in T$, sa couleur gauche est $\langle a]$ qui contient (b, a) où $b \in T$. Par construction, $a\mathcal{R}b$ si et seulement si $[a\rangle = \langle b]$. Un *jeu de tuiles de Wang* est donc un triplet $(T, \mathcal{H}, \mathcal{V})$ où T est un ensemble fini de tuiles et $\mathcal{H}$ et $\mathcal{V}$ sont des relations domino sur T. Deux tuiles t et t' sont compatibles horizontalement si $[t\rangle_{\mathcal{H}} = \langle t']_{\mathcal{H}}$. Deux tuiles t et t' sont compatibles verticalement si $[t\rangle_{\mathcal{V}} = \langle t']_{\mathcal{V}}$. Un *pavage* $\mathcal{T} \in T^{\mathbb{Z}^2}$ satisfait les contraintes locales, *i.e.* pour toute position $z \in \mathbb{Z}^2$, $\mathcal{T}(z)$ et $\mathcal{T}(z + (1, 0))$ sont compatibles horizontalement et $\mathcal{T}(z)$ et $\mathcal{T}(z + (0, 1))$ sont compatibles verticalement. En munissant $T^{\mathbb{Z}^2}$ de la topologie de Cantor, voir section 1.1.3, on observe que l'ensemble des pavages X_τ d'un jeu de tuiles τ est un sous-shift : c'est un compact invariant par translation. Un jeu de tuiles *pave le plan* s'il possède au moins un pavage. Un pavage $\mathcal{T} \in T^{\mathbb{Z}^2}$ est *périodique*, de période $u \in \mathbb{Z}^2$ si, pour tout $z \in \mathbb{Z}^2$, $\mathcal{T}(z + u) = \mathcal{T}(z)$. Un pavage est *bipériodique* s'il possède deux vecteurs de périodicité non colinéaires. Par un petit raisonnement combinatoire, on se convainc que tout jeu de tuiles qui possède un pavage périodique possède un pavage bipériodique. Un pavage est t-fini pour $t \in T$ s'il est constamment égal à t partout sauf sur un support fini. Par compacité, si un jeu de tuiles τ ne pave pas le plan, il existe un entier $n \in \mathbb{N}$ tel qu'il est impossible de construire un carré de tuiles de τ de côté n qui satisfasse les contraintes locales. Un jeu de tuiles est *apériodique* si tous ses pavages sont apériodiques.

Nous nous concentrons ici sur les problèmes de pavabilité. Le lecteur trouvera dans B. Grünbaum et G. C. Shephard [37] et C. Radin [79] une introduction générale aux pavages, par des tuiles de Wang mais aussi par des pièces géométriques plus générales. Pour une étude des propriétés structurelles des pavages, nous invitions le lecteur à consulter B. Durand [26] et A. Ballier *et al* [5].

Une fois introduites les définitions élémentaires, on peut définir les problèmes de pavabilité. La *pavabilité du plan* consiste, étant donné un jeu de tuiles, à décider si ce jeu de tuiles pave le plan. La *pavabilité du plan à origine contrainte* consiste, étant donné un jeu de tuiles τ et une tuile $t \in \tau$, à décider s'il existe un pavage de τ qui contient la tuile t en position 0. La *pavabilité périodique* consiste, étant donné un jeu de tuiles, à décider si

ce jeu de tuiles possède un pavage périodique. La *pavabilité finie* consiste, étant donné un jeu de tuiles τ et une tuile $t \in \tau$, à décider s'il existe un pavage t-fini de τ. La *pavabilité à diagonale contrainte* consiste, étant donné un jeu de tuiles τ et un sous-ensemble des tuiles $D \subseteq \tau$, à décider s'il existe un pavage du quart de plan $\mathbb{N}^2$ dont les tuiles sur la diagonale $x = y$ sont dans D. Toutes ces questions de pavabilité sont indécidables.

Certaines formes de pavabilité ont une démonstration d'indécidabilité assez directe. Ainsi, l'arrêt des machines de Turing se réduit, d'une part, à la pavabilité à origine contrainte, en forçant l'unique tête Turing dans l'état initial à être positionnée en 0, et d'autre part, à la pavabilité finie, en forçant par la bordure du pavage fini la tête dans l'état initial à apparaître en bas d'un rectangle de calcul. Le cas de la pavabilité à diagonale contrainte, dont l'indécidabilité permet à A. Kahr *et al* [44] de résoudre le problème initial étudié par H. Wang et J. R. Büchi, n'est pas beaucoup plus difficile. La réduction procède en forçant l'apparition dans le pavage de segments initiaux du calcul de la machine de Turing de longueurs non bornées. Voici une indication pour aider le lecteur à résoudre le problème de pavabilité de $\mathbb{N}^2$ à ligne contrainte (qui se réduit facilement à la pavabilité à diagonale contrainte) : considérer un jeu de quatre tuiles A, B, a, b où les contraintes locales imposent à droite de A et de a soit a soit B, à droite de B et de b soit b soit A, au-dessus de A soit A soit B et au-dessus de B, a et b soit a soit b et où les tuiles autorisées sur la première ligne sont uniquement les tuiles A et B.

Les cas de la pavabilité du plan et de la pavabilité périodique, qui sont étroitement liés comme le montrent Y. Gurevich et I. Koriakov [39] en établissant leur inséparabilité récursive, sont plus complexes. Comme nous l'avons vu, l'ensemble des jeux de tuiles qui ne pavent pas le plan est récursivement énumérable : il suffit de trouver une taille de carré qui n'est pas pavable. De même, l'ensemble des jeux de tuiles qui pavent périodiquement le plan est récursivement énumérable : il suffit de trouver un carré pavable dont les couleurs à gauche et à droite (resp. en haut et en bas) coïncident. Il s'agit donc de trouver des méthodes pour construire des jeux de tuiles apériodiques et de coupler le problème de l'arrêt d'une machine de Turing avec ce jeu de tuiles de sorte que le nouveau jeu de tuiles pave le plan si et seulement si la machine de Turing ne s'arrête pas sur l'entrée vide. De fait, l'étude de la pavabilité et celle des jeux de tuiles apériodiques se confondent. Le problème posé par H. Wang fût rapidement résolu par un de ses étudiants en thèse, R. Berger en 1966 :

Théorème 26 (R. Berger [8, 9])**.** *La pavabilité du plan est indécidable.*

R. Berger construit un énorme jeu de tuiles apériodique et démontre comment l'utiliser pour réduire le problème de l'arrêt des machines de Turing. Un résultat moins connu est qu'il exhibe en annexe de sa thèse [8] un jeu apériodique de 104 tuiles. La démonstration de R. Berger est longue et complexe et, en 1971, R. M. Robinson [84] propose une construction plus élégante d'un petit jeu de tuiles, une méthode de preuve d'apériodicité par composition, ainsi qu'une réduction du problème de l'arrêt plus simple reposant sur une technique d'ombrage détaillée plus loin. Si cet article reste la référence sur l'indécidabilité de la pavabilité du plan, la démonstration comporte des zones de flou où le travail combinatoire de vérification de la composition est laissé au lecteur. Le lecteur trouvera dans H. Wang [91] une discussion de ces résultats ainsi que des remarques structurelles sur la pavabilité par des tuiles de Wang.

À partir de ces résultats d'indécidabilité, l'étude des pavages a emprunté deux voies principales. D'une part, certains se sont intéressés à l'étude des pavages les plus complexes : ainsi W. P. Hanf [40] et D. Myers [71] établirent l'existence de jeux de tuiles qui pavent le plan mais dont tous les pavages sont non récursifs et B. Durand *et al* [28] construisirent, à l'aide de la complexité de Kolmogorov, des jeux de tuiles dont les pavages sont de complexité maximale. D'autre part, les intrigants jeux de tuiles apériodiques furent l'objet d'une compétition pour la détermination du plus petit jeu de tuiles apériodiques, dans l'espoir de mieux comprendre les mécanismes mettant à jour l'apériodicité. Pour un résumé de cette compétition et la description des jeux de tuiles associés, le lecteur consultera B. Grünbaum et G. C. Shephard [37, chap. 10 et 11] et l'actuel challenger, avec 13 tuiles, obtenu par K. Čulik [18].

L'utilisation de la pavabilité pour l'étude des propriétés des automates cellulaires, d'une part, nécessitant d'adapter les résultats de pavabilité et, d'autre part, plongeant des propriétés des pavages qu'il convient de comprendre dans la dynamique des automates cellulaires, nous avons besoin non seulement de la pavabilité comme d'un résultat boîte noire établi, mais aussi d'en comprendre les détails de la démonstration. Aussi, nous présentons ci-dessous trois méthodes de démonstration de l'indécidabilité de la pavabilité.

Méthode par composition

La première méthode, qui est la méthode historique, est la plus géométrique. Dans un premier temps, un jeu de tuiles apériodiques est construit par composition. Puis, dans un second temps, ce jeu de tuiles est modifié pour y ajouter le calcul Turing et rendre effective la réduction.

Cette méthode construit des jeux de tuiles apériodiques qui sont auto-similaires en faisant intervenir deux types d'objets. D'une part les tuiles avec leurs contraintes locales, d'autre part une substitution qui à une tuile associe un agencement de tuiles de sorte que l'ensemble des images des tuiles par cette substitution soit, en un certain sens, isomorphe au jeu de tuiles initial. La démonstration procède alors comme suit. Tout d'abord, on montre que toute tuile d'un pavage appartient à un agencement de tuiles et, récursivement, que tout agencement appartient à l'image d'un agencement par la substitution. Ainsi, toute tuile d'un pavage est dans l'image itérée de la substitution, qui est nécessairement apériodique si la substitution possède les bonnes propriétés syntaxiques. Dans cette version faible, qui est la méthode employée implicitement par R. M. Robinson [84], ou explicitement par R. Ammann *et al* [2] sur des tuiles polygonales, l'apériodicité est conséquence du fait que les pavages contiennent un objet lié à la substitution. B. Durand *et al* [27] ont proposé une construction simplifiée, avec une démonstration élégante, reposant sur ce principe pour des substitutions de tuiles de Wang par des carrés de tuiles 2×2.

Une fois le jeu de tuiles apériodiques obtenu par composition, la méthode d'ombrage, due à R. M. Robinson [84], consiste à utiliser le caractère auto-similaire de la substitution pour assurer la présence partout dans le pavage de zones de calcul de toute taille avec un coin identifié où initialiser le calcul. Chaque proto-tuile obtenue par substitution projette une ombre sur les tuiles de niveau inférieur. Si la substitution est bien choisie, l'ombrage définit des zones ensoleillées de toutes tailles présentes partout.

Nous avons proposé dans [C4] une nouvelle construction qui est une version forte de la méthode par composition, qui permet en outre de se passer de la méthode d'ombrage. Pour toute substitution $s : \Sigma \to \Sigma^{\boxplus}$ sur un alphabet fini Σ, soit $S : \Sigma^{\mathbb{Z}^2} \to \Sigma^{\mathbb{Z}^2}$ l'application de la substitution à un coloriage du plan, définie pour tout coloriage $C \in \Sigma^{\mathbb{Z}^2}$, tout $z \in \mathbb{Z}^2$ et tout $c \in \mathbb{Z}_2^2$ par $S(C)(2z + c) = s(C(z))(c)$. L'ensemble limite Λ_S de la substitution est le sous-shift non vide $\Lambda_S = \bigcap_{n\in\mathbb{N}} \Lambda_S^{(n)}$ où $\Lambda_S^{(0)} = \Sigma^{\mathbb{Z}^2}$ et $\Lambda_S^{(n+1)} = \left\{\sigma_z(\mathcal{C}) \middle| \mathcal{C} \in S(\Lambda_S^{(n)}), z \in \mathbb{Z}^2\right\}$. Après avoir donné une condition syntaxique simple sur s pour l'apériodicité de Λ_S, nous construisons une substitution apériodique s et un jeu de 104 tuiles dont les contraintes locales forcent son ensemble de pavages, non vide, à être un sous-ensemble de Λ_S. Puis, nous montrons comment modifier ce jeu de tuiles pour transformer toute substitution t en un jeu de tuiles τ dont un quotient de l'ensemble des pavages X_τ est égal à Λ_t, *i.e.* $\overline{\pi}(X_\tau) = \Lambda_t$ où π est un coloriage des tuiles de τ dans les lettres de t. Ainsi, l'ombrage peut être remplacé par une substitution dont l'ensemble limite contient partout des zones de calcul de toutes tailles avec un coin pointé.

Méthode par point fixe

Indépendamment de nos travaux, B. Durand *et al* [30] ont mis au point récemment une nouvelle méthode pour démontrer l'indécidabilité de la pavabilité. Cette méthode se distingue des autres par le fait qu'elle n'est pas du tout géométrique et propose une approche plus calculatoire. L'idée consiste à utiliser un analogue du théorème du point fixe de Kleene, voir P. G. Odifreddi [75]. À tout triplet, constitué de deux entiers n et k et d'une machine de Turing M, on associe un jeu de tuiles τ de sorte que tout pavage par ce jeu de tuiles se décompose en une grille régulière de carrés de côté n qui échangent avec leurs quatre voisins k bits d'information sur chaque bord et qui exécutent la machine M sur l'entrée constituée des $4k$ bits de sorte qu'un bloc n'apparaisse dans le pavage que si l'entrée est reconnue par la machine en temps inférieur à n. Le jeu de tuiles simule donc, en un certain sens, le jeu de tuiles codé par la machine M. Sous de bonnes hypothèses, le codage possède un point fixe : un jeu de tuiles dont tout pavage est constitué d'une grille régulière de blocs qui codent des tuiles de M. Ce jeu de tuiles, qui code une substitution, est nécessairement apériodique. L'approche peut être affinée pour coder via les proto-tuiles la réduction du problème de l'arrêt des machines de Turing.

Méthode par transducteur et mots sturmiens

Une troisième méthode obtenue indépendamment par J. Kari [51] repose sur ses travaux initiaux pour la construction de petits jeux de tuiles apériodiques, dont le jeu de 13 tuiles de K. Čulik [18] est une optimisation. Le principe consiste à considérer une tuile de Wang comme un élément d'un transducteur qui force une relation entre le mot biinfini constitué par la ligne des couleurs nord et le mot biinfini constitué par la ligne des couleurs sud : les couleurs ouest et est désignent les états du transducteur. Le second ingrédient de la construction consiste à remarquer que par un codage sturmien, les mots biinfinis peuvent être utilisés pour coder des tuples de nombres réels sur lesquels les tuiles peuvent effectuer certaines transformations affines. Ainsi, J. Kari réduit le problème de l'immortalité des systèmes de transformations affines à la pavabilité du plan. Or, l'immortalité des machines de Turing, qui est indécidable comme nous le verrons en section 2.3, se réduit à l'immortalité des systèmes de transformations affines. La démonstration obtenue est alors courte

et élégante. Par rapport à l'approche par composition, elle perd une certaine flexibilité qui permet les adaptations nécessaires aux réductions sur automates cellulaires.

Problème ouvert 10. *Adapter les techniques classiques de réduction aux automates cellulaires à la méthode par transducteur et mots sturmiens.*

2.2.2 Application aux automates cellulaires

La pavabilité est un outil pour obtenir l'indécidabilité de propriétés des automates cellulaires. Ainsi, J. Kari [45] obtient l'indécidabilité de l'injectivité des automates cellulaires en dimension 2 en réduisant la pavabilité ; et l'indécidabilité de la surjectivité des automates cellulaires en dimension 2 en réduisant la pavabilité finie. Le principe général est le suivant : la configuration de l'automate cellulaire comporte une couche de tuiles de Wang et un fil le long duquel s'effectue une addition dans $\mathbb{Z}_2$ si le pavage est localement valide, de sorte que l'automate cellulaire soit injectif si et seulement si il n'existe aucun pavage valide du plan (l'injectivité nécessite une couche de pavage supplémentaire pour garantir un fil raisonnable qui remplit des zones denses du plan, la surjectivité est traitée grâce à son équivalence avec l'injectivité sur les configurations finies).

Pour relier les pavages aux automates cellulaires de dimension 1, on remarque, d'une part, que, par groupage, tout automate cellulaire est équivalent à un automate cellulaire de voisinage $\{0, 1\}$ et, d'autre part, que l'ensemble limite d'un automate cellulaire avec un tel voisinage est exactement l'ensemble des pavages par un jeu de tuiles NW-déterministe. Un jeu de tuiles est NW-déterministe si, pour toute paire de couleurs il existe au plus une tuile qui possède ce couple de couleurs sur ses côtés nord et ouest. Après avoir réduit la pavabilité à la pavabilité pour des jeux de tuiles NW-déterministes, J. Kari [47] obtient l'indécidabilité de la nilpotence en réduisant la pavabilité pour des jeux de tuiles NW-déterministes à la non nilpotence. À tout jeu de tuiles NW-déterministe, il associe un automate cellulaire dont les états sont les tuiles auxquelles s'ajoute une tuile d'erreur. La règle locale de transition consiste à associer à toute paire de tuiles l'unique tuile qu'elles encadrent si celle-ci existe ou la tuile d'erreur, qui se propage, sinon. Si le jeu de tuiles pave le plan, toute diagonale d'un pavage est dans l'ensemble limite. Réciproquement, si le jeu de tuiles ne pave pas le plan, il existe un côté de carré que le jeu de tuiles ne pave pas, c'est une borne sur le temps au bout duquel l'automate cellulaire devient nilpotent.

Fort de ce résultat, J. Kari [49] le généralise en un théorème de Rice sur les ensembles limites : étant donné un ensemble non trivial d'ensembles limites, savoir si l'ensemble limite d'un automate cellulaire donné est dans cet ensemble est indécidable. On notera que l'ensemble des états de l'automate cellulaire considéré n'est pas fixé.

Un certain nombre de variantes sont obtenues par la suite. Ainsi, I. Rapaport et J. Mazoyer [64] obtiennent l'indécidabilité de la nilpotence sur les configurations périodiques en réduisant la pavabilité périodique à la pavabilité périodique pour des jeux de tuiles NW-déterministes.

Dans [C3], nous avons utilisé ce dernier résultat pour obtenir l'indécidabilité de l'universalité intrinsèque, voir section 1.3.3.

2.2.3 Pavabilité par des polyominos

Nous nous sommes intéressés plus récemment au pavage du plan par des polyominos. Alors que le pavage par des tuiles de Wang s'intéresse à des carrés unité colorés, le pavage par des polyominos est un pavage par des formes. Un *polyomino* est l'union simplement connexe d'un nombre fini de carrés unité. Un jeu de polyominos de taille k est un ensemble de k polyominos. Un pavage par un jeu de polyominos est une partition du plan de sorte que chaque région soit obtenue comme isométrie d'un des polyominos du jeu. Un pavage par translation par un jeu de polyominos est un pavage dans lequel seules les translations sont autorisées (ni rotation ni réflexion).

Les problèmes de pavabilité par des polyominos ont été étudiés par S. W. Golomb [35]. En particulier, le lecteur y trouvera une réduction d'un jeu de n tuiles de Wang en un jeu de n polyominos qui préserve la pavabilité, ce qui assure l'indécidabilité de la pavabilité par translation d'un jeu de polyominos et, si on prend soin aux détails, l'indécidabilité de la pavabilité par un jeu de polyominos. Nous nous sommes intéressés aux questions de pavabilité par un jeu de polyominos de taille fixée.

Dans le cadre de la pavabilité par translation, H. A. G. Wijshoff et J. van Leeuwen [92] on montré que le cas d'un unique polyomino est décidable. Ce résultat a été affiné en une caractérisation complète des pavages et une caractérisation syntaxique des polyominos qui pavent le plan, les pseudo-hexagones, par D. Beauquier et M. Nivat [7]. Plus récemment, I. Gambini et L. Vuillon [34] ont mis au point un algorithme quadratique en la taille du polyomino pour tester sa pavabilité par translation. Dans [U6] nous réduisons tout jeu de tuiles de Wang à un jeu de 11 polyominos en préservant la pavabilité ; et une étude attentive de R. Ammann *et al* [2] nous permet de construire un jeu de 8 polyominos apériodique par translation.

Théorème 27 ([U6])**.** *La pavabilité par translation par un jeu de 11 polyominos est indécidable.*

Problème ouvert 11. *La pavabilité par translation est-elle décidable pour les jeux de k polyominos, $2 \leqslant k \leqslant 10$?*

Dans le cadre de la pavabilité par isométries, la décidabilité et l'apériodicité sont toujours ouvertes pour le cas d'un unique polyomino. Nous avons tiré parti des isométries pour réduire notre jeu à 5 polyominos et construire à partir de R. Ammann *et al* [2] un jeu de 3 polyominos apériodique.

Théorème 28 ([U6])**.** *La pavabilité par un jeu de 5 polyominos est indécidable.*

Problème ouvert 12. *La pavabilité est-elle décidable pour les jeux de k polyominos, $1 \leqslant k \leqslant 4$.*

2.3 Mortalité et périodicité

Notre second domaine de contribution à l'étude de la décidabilité de propriétés dynamiques des systèmes complexes concerne l'étude des propriétés de mortalité et de périodicité des machines de Turing. Introduit par J. R. Büchi en même temps que le problème

de pavabilité, dans le but de résoudre un même problème originel, l'immortalité des machines de Turing est indécidable. P. K. Hooper [42], un étudiant en thèse de H. Wang, obtient ce résultat par réduction de l'arrêt des machines de Turing. En adaptant ce résultat au cas des machines de Turing réversibles, nous avons montré avec J. Kari [C6, U5] que la périodicité des automates cellulaires est indécidable. Il s'agit du premier résultat d'indécidabilité de propriétés dynamiques des automates cellulaires qui ne réduisent pas la pavabilité. Le lecteur trouvera les détails techniques dans [U5] reproduit en annexe B.3.

2.3.1 Immortalité des machines de Turing

La réversibilité des machines de Turing se caractérise syntaxiquement. L'inverse d'une machine de Turing est la machine de Turing dont la table de transition est constituée des instructions inverses. L'inverse $(s, \delta, t)^{-1}$ d'une instruction de *mouvement* (s, δ, t) d'une machine de Turing est l'instruction (t, δ^{-1}, s) où δ^{-1} est la direction inverse de δ. L'inverse $(s, a, t, b)^{-1}$ d'une instruction de *lecture/écriture* (s, a, t, b) est l'instruction (t, b, s, a). L'inverse $(S, \Sigma, T)^{-1}$ d'une machine de Turing (S, Σ, T) est la machine de Turing (S, Σ, T^{-1}) où $T^{-1} = \{\iota^{-1} | \iota \in T\}$. Une machine de Turing *réversible* est une machine de Turing déterministe d'inverse déterministe.

Une configuration d'une machine de Turing est mortelle si son orbite est finie, c'est-à-dire si, partant de cette configuration, la machine finit par s'arrêter. Une machine de Turing dont toutes les configurations sont mortelles est *mortelle*. Une machine de Turing qui possède au moins une configuration immortelle est *immortelle*. Par compacité, la fonction globale d'une machine de Turing étant continue et l'ensemble des configurations d'arrêt formant un ouvert, une machine de Turing mortelle est uniformément mortelle : il existe une borne T telle que, au bout d'au plus T étapes de calcul, toute configuration s'arrête. Le problème de l'immortalité consiste, étant donné une machine de Turing, à décider si elle est immortelle. Les machines de Turing qui possèdent une orbite infinie périodique sont récursivement énumérables, de même que celles qui sont mortelles. Aussi, l'indécidabilité est ici aussi engendrée par les configurations apériodiques.

La démonstration originelle de P. K. Hooper [42] procède par réduction du problème de l'arrêt des machines à deux compteurs en utilisant une astucieuse construction récursive dans laquelle la machine de Turing construite utilise le ruban à la fois comme ruban de calcul mais aussi comme espace de pile pour effectuer des appels récursifs. La démonstration est très technique.

Théorème 29 (P. K. Hooper [42])**.** *L'immortalité des machines de Turing est indécidable.*

Afin d'obtenir notre résultat sur la périodicité des automates cellulaires, nous avions besoin d'une propriété plus forte : l'indécidabilité de l'immortalité des machines de Turing réversibles. Aussi, nous avons dû reprendre la démonstration de Hooper et l'adapter au cas réversible. À cette fin, nous avons introduit un langage de programmation de machines de Turing réversibles (Gnirut, disponible à l'adresse `http://www.lif.univ-mrs.fr/~nollinge/rec/gnirut/`), afin d'obtenir une preuve plus robuste, dans laquelle la réduction est explicitement décrite. Notre démonstration, qui se substitue à celle de Hooper procède par réduction du problème de l'arrêt des machines réversibles à deux compteurs.

Théorème 30 ([C6, U5]). *L'immortalité des machines de Turing réversibles est indécidable.*

Ce résultat nous a permis d'obtenir un nouveau théorème d'indécidabilité sur la périodicité des machines de Turing. Une machine de Turing est complète si toutes ses transitions sont spécifiées, c'est-à-dire si sa fonction globale est totale. Une machine de Turing est *périodique* si elle est complète et que toutes ses orbites sont périodiques. Les machines de Turing périodiques sont réversibles. Par compacité, la fonction globale d'une machine de Turing étant continue et l'ensemble des configurations périodiques de période fixée formant un ouvert, une machine de Turing périodique est uniformément périodique : il existe une période uniforme P telle que toutes les configurations sont périodiques de période P. Les machines de Turing périodiques dans notre modèle avec le ruban qui se déplace sont les mêmes que les machines de Turing périodiques dans le modèle classique où la tête se déplace. Le problème de périodicité consiste, étant donné une machine de Turing, à décider si elle est périodique.

Théorème 31 ([C6, U5]). *La périodicité des machines de Turing est indécidable.*

2.3.2 Application aux automates cellulaires

Forts de ce résultat sur la périodicité des machines de Turing, nous pouvons clore notre chaîne de réduction. Étant donné une machine de Turing réversible complète, nous construisons un automate cellulaire qui simule cette machine de Turing et son inverse. Chaque cellule contient une lettre du ruban et une information sur la tête : soit son état et le sens, direct ou inverse, s'il y a une tête sur cette case, soit une flèche qui indique la direction dans laquelle se trouve la tête. Les têtes réalisent une simulation de la machine de Turing codée tant que les flèches et têtes qu'elles observent sont cohérentes. En cas d'incohérence, la tête inverse son sens de simulation. L'automate cellulaire ainsi construit est périodique si et seulement si la machine de Turing simulée est périodique.

Théorème 32 ([C6, U5]). *La périodicité des automates cellulaires est indécidable.*

2.3.3 Poursuite des travaux

L'utilisation de l'indécidabilité pour mettre en avant la complexité de certaines propriétés dynamiques des automates cellulaires n'en est qu'à ses balbutiements. Les résultats des années 90 de J. Kari en sont l'amorce. Ce nouveau résultat sur la périodicité est encourageant, dans le sens où il donne de nouveaux outils et montre l'indécidabilité d'une propriété extrêmement naturelle qui semble *a priori* peu calculatoire (de fait, la conjecture était que la périodicité était décidable en dimension 1). Il s'agit maintenant de poursuivre cet effort en s'attaquant de front aux classifications topologiques.

Problème ouvert 13. *Démontrer que la plupart des propriétés dynamiques issues de classifications de dynamique topologique sont indécidables.*

BIBLIOGRAPHIE

[1] J. Albert et K. Čulik, II, A simple universal cellular automaton and its one-way and totalistic version, *Complex Systems*, **1**, n° 1 (1987) 1–16.

[2] R. Ammann, B. Grünbaum et G. Shephard, Aperiodic tiles, *Discrete and Computational Geometry*, **8**, n° 1 (1992) 1–25.

[3] S. Amoroso et Y. N. Patt, Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures, *Journal of Computer and System Sciences*, **6** (1972) 448–464.

[4] J. T. Baldwin et S. Shelah, On the classifiability of cellular automata, *Theoretical Computer Science*, **230**, n° 1-2 (2000) 117–129.

[5] A. Ballier, B. Durand et E. Jeandel, Structural aspects of tilings, *STACS*, edité par S. Albers et P. Weil, volume 08001 de *Dagstuhl Seminar Proceedings* (Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008), (p. 61–72).

[6] E. R. Banks, Universality in cellular automata, *Symposium on Switching and Automata Theory (Santa Monica, California, 1970)* (IEEE, 1970), (p. 194–215).

[7] D. Beauquier et M. Nivat, On translating one polyomino to tile the plane, *Discrete and Computational Geometry*, **6**, n° 1 (1991) 575–592.

[8] R. Berger, *The Undecidability of the Domino Problem*, Thèse de doctorat, Harvard University, 1964.

[9] R. Berger, The Undecidability of the Domino Problem, *Memoirs American Mathematical Society*, **66**.

[10] E. R. Berlekamp, J. H. Conway et R. K. Guy, *Winning ways for your mathematical plays. Vol. 2* (Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982), games in particular.

[11] J. L. Britton, The Word Problem, *The Annals of Mathematics, Second Series*, **77**, n° 1 (1963) 16–32.

[12] J. R. Büchi, Turing-Machines and the Entscheidungsproblem, *Mathematische Annalen*, **148** (1962) 201–213.

[13] J. Cervelle, Complexité dynamique et algorithmique des automates cellulaires, 2007, habilitation à diriger des recherches, Université Paris-Est Marne-la-Vallée.

[14] V. Claus, Some remarks on PCP(k) and related problems, *Bulletin of the EATCS*, **12** (1980) 54–61.

[15] E. F. Codd, *Cellular Automata* (Academic Press, New York, 1968).

[16] J. H. Conway, *Regular Algebra and Finite Machines* (Chapman Hall, 1971).

[17] M. Cook, Universality in Elementary Cellular Automata, *Complex Systems*, **15** (2004) 1–40.

[18] K. Čulik, II, An aperiodic set of 13 Wang tiles, *Discrete Mathematics*, **160**, n° 1-3 (1996) 245–251.

[19] K. Čulik, II, L. P. Hurd et S. Yu, Computation theoretic aspects of cellular automata, *Physica D*, **45** (1990) 357–378.

[20] K. Čulik, II, J. Pachl et S. Yu, On the limit sets of cellular automata, *SIAM Journal on Computing*, **18**, n° 4 (1989) 831–842.

[21] K. Čulik, II et S. Yu, Undecidability of CA classification schemes, *Complex Systems*, **2** (1988) 177–190.

[22] M. D. Davis, A note on universal Turing machines, *Automata Studies*, edité par C. E. Shannon et J. McCarthy (Princeton University Press, Princeton, 1956), (p. 167–175).

[23] M. Delorme, An introduction to cellular automata: some basic definitions and concepts, *Cellular automata (Saissac, 1996)*, edité par M. Delorme et J. Mazoyer (Kluwer Acad. Publ., Dordrecht, 1999), (p. 5–49).

[24] M. Delorme et J. Mazoyer, Cellular automata as languages recognizers, *Cellular automata (Saissac, 1996)* (Kluwer Acad. Publ., Dordrecht, 1999), (p. 153–179).

[25] B. Durand, *Cellular automata: reversibility and complexity*, Thèse de doctorat, École Normale Supérieure de Lyon, 1994.

[26] B. Durand, Tilings and Quasiperiodicity, *Theoretical Computer Science*, **221**, n° 1-2 (1999) 61–75.

[27] B. Durand, L. Levin et A. Shen, Local rules and global order, or aperiodic tilings, *Math. Intelligencer*, **27**, n° 1 (2005) 64–68.

[28] B. Durand, L. A. Levin et A. Shen, Complex tilings, *STOC* (2001), (p. 732–739).

[29] B. Durand et Z. Róka, The game of life: universality revisited, *Cellular automata (Saissac, 1996)*, edité par M. Delorme et J. Mazoyer (Kluwer Acad. Publ., Dordrecht, 1999), (p. 51–74).

[30] B. Durand, A. Romashchenko et A. Shen, Fixed point and aperiodic tilings, 2007, *preprint*.

[31] Y. G. Ergon Börger, Erich Grädel, *The classical decision problem* (Springer-Verlag Telos, 1996).

[32] P. C. Fischer, Generation of Primes by a One-Dimensional Real-Time Iterative Array, *Journal of the ACM*, **12**, n° 3 (1965) 388–394.

[33] E. Formenti, *Cellular automata and chaos: from topology to Kolmogorov complexity*, Thèse de doctorat, École Normale Supérieure de Lyon, 1998.

[34] I. Gambini et L. Vuillon, An algorithm for deciding if a polyomino tiles the plane, *Theoretical Informatics and Applications*, **41**, n° 2 (2007) 147–155.

[35] S. W. Golomb, Tiling with sets of polyominoes, *Journal of Combinatorial Theory*, **9**, n° 1 (1970) 60–71.

[36] R. W. Gosper, Exploiting regularities in large cellular spaces, *Physica D : Nonlinear Phenomena*, **10**, n° 1-2 (1984) 75–80.

[37] B. Grünbaum et G. C. Shephard, *Tilings and patterns*, A Series of Books in the Mathematical Sciences (W. H. Freeman and Company, New York, 1989), an introduction.

[38] P. Guillon et G. Richard, Nilpotency and Limit Set of Cellular Automata, *Mathematical Foundations of Computer Science (MFCS'2008)*, edité par E. Ochmański et J. Tyszkiewicz, volume 5162 de *Lecture Notes in Computer Science* (Springer, Berlin, 2008), (p. 375–386).

[39] Y. Gurevich et I. Koriakov, A remark on Berger's paper on the domino problem, *Siberian Journal of Mathematics*, **13** (1972) 459–463, (in Russian).

[40] W. P. Hanf, Nonrecursive Tilings of the Plane I, *Journal of Symbolic Logic*, **39**, n° 2 (1974) 283–285.

[41] G. A. Hedlund, Endormorphisms and automorphisms of the shift dynamical system, *Mathematical Systems Theory*, **3** (1969) 320–375.

[42] P. K. Hooper, The Undecidability of the Turing Machine Immortality Problem, *Journal of Symbolic Logic*, **31**, n° 2 (1966) 219–234.

[43] W. Hordijk, C. R. Shalizi et J. P. Crutchfield, Upper bound on the products of particle interactions in cellular automata, *Physica D : Nonlinear Phenomena*, **154**, n° 3-4 (2001) 240–258.

[44] A. S. Kahr, E. F. Moore et H. Wang, Entscheidungsproblem reduced to the $\forall\exists\forall$ case, *Proc. Natl. Acad. Science*, **48**, n° 3 (1962) 365–377.

[45] J. Kari, Reversibility of 2D cellular automata is undecidable, *Physica D. Nonlinear Phenomena*, **45**, n° 1-3 (1990) 379–385, cellular automata: theory and experiment (Los Alamos, NM, 1989).

[46] J. Kari, The nilpotency problem of one-dimensional cellular automata, *SIAM Journal on Computing*, **21**, n° 3 (1992) 571–586.

[47] J. Kari, The nilpotency problem of one-dimensional cellular automata, *SIAM J. Comput.*, **21**, n° 3 (1992) 571–586.

[48] J. Kari, Reversibility and surjectivity problems of cellular automata, *Journal of Computer and System Sciences*, **48**, n° 1 (1994) 149–182.

[49] J. Kari, Rice's theorem for the limit sets of cellular automata, *Theoretical Computer Science*, **127**, n° 2 (1994) 229–254.

[50] J. Kari, Theory of cellular automata : A survey, *Theoretical Computer Science*, **334** (2005) 3–33.

[51] J. Kari, The Tiling Problem Revisited (Extended Abstract), *MCU 2007*, volume 4664 de *LNCS* (Springer, 2007), (p. 72–79).

[52] P. Kůrka, Languages, equicontinuity and attractors in cellular automata, *Ergodic Theory and Dynamical Systems*, **17** (1997) 417–433.

[53] P. Kůrka, On Topological Dynamics of Turing Machines, *Theoretical Computer Science*, **174**, n° 1-2 (1997) 203–216.

[54] M. Kunc, The Power of Commuting with Finite Sets of Words, *Theory of Computing Systems*, **40**, n° 4 (2007) 521–551.

[55] R. E. Ladner, The circuit value problem is log space complete for P, *SIGACT News*, **7**, n° 1 (1975) 18–20.

[56] C. Langton, Self-Reproduction in Cellular Automata, *Physica D*, **10**, n° 1-2 (1984) 135–144.

[57] D. Lind et B. Marcus, *An introduction to symbolic dynamics and coding* (Cambridge University Press, Cambridge, 1995).

[58] K. Lindgren et M. G. Nordahl, Universal computation in simple one-dimensional cellular automata, *Complex Systems*, **4**, n° 3 (1990) 299–318.

[59] O. Martin, A. M. Odlyzko et S. Wolfram, Algebraic properties of cellular automata, *Communications in Mathematical Physics*, **93**, n° 2 (1984) 219–258.

[60] A. Maruoka et M. Kimura, Condition for Injectivity of Global Maps for Tessellation Automata, *Information and Control*, **32** (1976) 158–162.

[61] Y. Matiyasevich et G. Sénizergues, Decision problems for semi-Thue systems with a few rules, *Theoretical Computer Science*, **330**, n° 1 (2005) 145–169.

[62] J. Mazoyer, A six-state minimal time solution to the firing squad synchronization problem, *Theoretical Computer Science*, **50**, n° 2 (1987) 183–240.

[63] J. Mazoyer et I. Rapaport, Additive cellular automata over Z_p and the bottom of $(\mathrm{CA}, \leq)$, *Mathematical foundations of computer science (Brno, 1998)* (Springer, Berlin, 1998), (p. 834–843).

[64] J. Mazoyer et I. Rapaport, Global fixed point attractors of circular cellular automata and periodic tilings of the plane: undecidability results, *Discrete Mathematics*, **199**, n° 1-3 (1999) 103–122.

[65] J. Mazoyer et I. Rapaport, Inducing an order on cellular automata by a grouping operation, *Discrete Applied Mathematics*, **91**, n° 1-3 (1999) 177–196.

[66] J. Mazoyer et V. Terrier, Signals in one-dimensional cellular automata, *Theoretical Computer Science*, **217**, n° 1 (1999) 53–80, cellular automata (Milan, 1996).

[67] M. L. Minsky, Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in Theory of Turing Machines, *Annals of Mathematics*, **74**, n° 3 (1961) 437–455.

[68] M. L. Minsky, *Computation : Finite and Infinite Machines* (Prentice Hall, Englewoods Cliffs, 1967).

[69] E. F. Moore, Machine models of self-reproduction, *Proceedings of Symposia in Applied Mathematics*, volume 14 (American Mathematical Society, 1962), (p. 17–33).

[70] E. F. Moore, The Firing Squad Synchronization Problem, *Sequential Machines - Selected Papers*, edité par E. F. Moore (Addison-Wesley, 1964), (p. 213–214).

[71] D. Myers, Nonrecursive Tilings of the Plane II, *Journal of Symbolic Logic*, **39**, n° 2 (1974) 286–294.

[72] J. Myhill, The converse of Moore's Garden-of-Eden theorem, *Proceedings of the American Mathematical Society*, volume 14 (American Mathematical Society, 1963), (p. 658–686).

[73] T. Neary et D. Woods, P-completeness of cellular automaton Rule 110, *Proceedings of ICALP 2006*, volume 4051 de *Lecture Notes in Computer Science* (Springer, Berlin, 2006), (p. 132–143).

[74] J. von Neumann, *Theory of Self-Reproducing Automata* (University of Illinois Press, Urbana, Ill., 1966).

[75] P. G. Odifreddi, *Classical Recursion Theory* (Elsevier, The Netherlands, 1989).

[76] E. L. Post, *The Two-Valued Iterative Systems of Mathematical Logic* (Princeton University Press, Princeton, 1941).

[77] E. L. Post, Recursively Enumerable Sets of Positive Integers and their Decision Problems, *Bulletin of the AMS*, **50** (1944) 284–316.

[78] E. L. Post, A Variant of a Recursively Unsolvable Problem, *Bulletin of the American Mathematical Society*, **52** (1946) 264–268.

[79] C. Radin, *Miles of tiles*, volume 1 de *Student Mathematical Library* (American Mathematical Society, Providence, RI, 1999).

[80] I. Rapaport, *Inducing an order on cellular automata by a grouping operation*, Thèse de doctorat, École Normale Supérieure de Lyon, 1998.

[81] H. G. Rice, Classes of recursively enumerable sets and their decision problems, *Transactions of the AMS*, **74**, n° 2 (1953) 358–366.

[82] G. Richard, Rule 110 : Universality and Catenations, *Symposium on Cellular Automata Journées Automates Cellulaires (JAC'2008)*, edité par B. Durand (MCCME Publishing House, Moscow, 2008), (p. 141–160).

[83] D. Richardson, Tessellations with local transformations, *Journal of Computer and System Sciences*, **6** (1972) 373–388.

[84] R. M. Robinson, Undecidability and Nonperiodicity for Tilings of the plane, *Inventiones Mathematicae*, **12** (1971) 177–209.

[85] H. Rogers, Jr., Gödel Numberings of Partial Recursive Functions, *The Journal of Symbolic Logic*, **23**, n° 3 (1958) 331–341.

[86] H. Rogers, Jr., *The Theory of Recursive Functions and Effective Computability* (MIT Press, 1967).

[87] A. R. Smith, III, Real-time language recognition by one-dimensional cellular automata, *Journal of Computer and System Sciences*, **6** (1972) 233–253.

[88] G. Theyssier, *Automates cellulaires : un modèle de complexités*, Thèse de doctorat, École Normale Supérieure de Lyon, 2005.

[89] A. M. Turing, On computable numbers with an application to the Entscheidungs problem, *Proceedings of the London Mathematical Society 2*, **42** (1936) 230–265.

[90] H. Wang, Proving theorems by pattern recognition II, *Bell System Technical Journal*, **40** (1961) 1–42.

[91] H. Wang, Notes on a Class of Tiling Problems, *Fundamenta Mathematicae*, **82** (1975) 295–305.

[92] H. A. G. Wijshoff et J. van Leeuwen, Arbitrary versus periodic storage schemes and tessellations of the plane using one type of polyomino, *Information and Control*, **62**, n° 1 (1984) 1–25.

[93] D. A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models : An Introduction*, volume 1725 de *Lecture Notes in Mathematics* (Springer).

[94] S. Wolfram, Computation theory of cellular automata, *Communications in Mathematical Physics*, **96**, n° 1 (1984) 15–57.

[95] S. Wolfram, *A New Kind of Science* (Wolfram Media, Inc., Champaign, Ill., 2002).

[96] D. Woods et T. Neary, The Complexity of Small Universal Turing Machines, *Computability in Europe*, edité par S. B. Cooper, B. Löwe et A. Sorbi, volume 4497 de *Lecture Notes in Computer Science* (Springer, 2007), (p. 791–799).

[97] K. Zuse, Rechnender Raum, *Elektronische Datenverarbeitung*, **8** (1967) 336–344.

PUBLICATIONS

Thèse

[T1] N. Ollinger, *Automates cellulaires : structures*, Thèse de doctorat, École Normale Supérieure de Lyon, 2002.

Conférences internationales avec comité de relecture

[C1] N. Ollinger, Two-states bilinear intrinsically universal cellular automata, *Fundamentals of computation theory (FCT'2001)*, edité par R. Freivalds, volume 2138 de *Lecture Notes in Computer Science* (Springer, Berlin, 2001), (p. 396–399).

[C2] N. Ollinger, The quest for small universal cellular automata, *International Colloquium on Automata, languages and programming (ICALP'2002)*, edité par P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz et R. Conejo, volume 2380 de *Lecture Notes in Computer Science* (Springer, Berlin, 2002), (p. 318–329).

[C3] N. Ollinger, The intrinsic universality problem of one-dimensional cellular automata, *Symposium on Theoretical Aspects of Computer Science (STACS'2003)*, edité par H. Alt et M. Habib, volume 2607 de *Lecture Notes in Computer Science* (Springer, Berlin, 2003), (p. 632–641).

[C4] N. Ollinger, Two-by-two Substitution Systems and the Undecidability of the Domino Problem, *Computability in Europe (CiE'2008)*, edité par A. Beckmann, C. Dimitracopoulos et B. Löwe, volume 5028 de *Lecture Notes in Computer Science* (Springer, Berlin, 2008), (p. 476–485).

[C5] N. Ollinger, Universalities in Cellular Automata : a (short) survey, *Symposium on Cellular Automata Journées Automates Cellulaires (JAC'2008)*, edité par B. Durand (MCCME Publishing House, Moscow, 2008), (p. 102–118).

[C6] J. Kari et N. Ollinger, Periodicity and Immortality in Reversible Computing, *Mathematical Foundations of Computer Science (MFCS'2008)*, edité par E. Ochmański et J. Tyszkiewicz, volume 5162 de *Lecture Notes in Computer Science* (Springer, Berlin, 2008), (p. 419–430).

[C7] N. Ollinger et G. Richard, Collisions and their Catenations : Ultimately Periodic Tilings of the Plane, *Proceedings of the Fifth IFIP Int. Conf. on TCS (IFIP-TCS'2008)*, volume 273/2008 (Springer, Boston, 2008), (p. 229–240).

Articles de revues avec comité de relecture

[J1] C. Choffrut, J. Karhumäki et N. Ollinger, The commutation of finite sets : a challenging problem, *Theoretical Computer Science*, **273** (2002) 69–79.

[J2] E. Jeandel et N. Ollinger, Playing with Conway's Problem, *Theoretical Computer Science*, (to appear doi :10.1016/j.tcs.2008.09.026).

Chapitres de livres

[L1] N. Ollinger, Universalities in Cellular Automata, *Handbook of Natural Computing*, edité par G. Rozenberg, T. Baeck et J. Kok (Springer, Berlin), (to appear).

Thèses encadrées

[P1] V. Bernardi, *Lois de conservation sur automates cellulaires*, Thèse de doctorat, Université de Provence, 2007, (co-encadré avec B. Durand).

[P2] G. Richard, *Systèmes de particules et collisions discrètes dans les automates cellulaires*, Thèse de doctorat, Université de Provence, 2008, (en cours).

Mémoires de Master encadrés

[M1] P. Guillon, *Calcul fiable en présence d'erreurs*, Mémoire de Master, École Normale Supérieure de Lyon, 2004, (co-encadré avec B. Durand).

[M2] G. Richard, *Auto-organisation et complexité : une approche par les automates cellulaires*, Mémoire de Master, École Normale Supérieure de Lyon, 2005.

[M3] F. Richoux, *Automates cellulaires : particules, collisions et cartes planaires*, Mémoire de Master, Université de Provence, 2005.

Travaux soumis et en cours d'écriture

[U1] M. Delorme, J. Mazoyer, N. Ollinger et G. Theyssier, Bulking I : an Abstract Theory of Bulking, (24pp).

[U2] M. Delorme, J. Mazoyer, N. Ollinger et G. Theyssier, Bulking II : Classifications of Cellular Automata, (36pp).

[U3] N. Ollinger et G. Richard, A Particular Universal Cellular Automaton, (14pp).

[U4] N. Ollinger et G. Richard, Automata on the Plane vs Particles and Collisions, (11pp).

[U5] J. Kari et N. Ollinger, Periodicity and Immortality in Reversible Computing, (28pp).

[U6] N. Ollinger, Tiling the Plane with a Fixed Number of Polyominoes, (10pp).

ANNEXE

A AUTOMATES CELLULAIRES, GÉOMÉTRIE ET CALCUL

Recueil des articles les plus représentatifs concernant le premier chapitre du mémoire. Le lecteur y trouvera les principaux résultats des trois principales sections : groupage, universalités et systèmes de particules et collisions. Lorsqu'elles existent, nous avons choisi de présenter les versions longues (éventuellement soumises mais non encore acceptées) plutôt que les communications de conférence correspondantes.

Sommaire

A.1 Bulking I : an Abstract Theory of Bulking

Version courante de [U1], première partie d'une série de deux articles écrite en collaboration avec M. Delorme, J. Mazoyer and G. Theyssier. Cet article et le suivant font la synthèse des travaux sur le groupage réalisés à Lyon à partir de la thèse fondatrice d'I. Rapaport [80] dont les principaux résultats ont été publiés par J. Mazoyer et I. Rapaport dans [63] et [65]. Plus précisément, ils synthétisent les résultats présentés en français dans les thèse de N. Ollinger [T1] et G. Theyssier [88].

Bulking I: an Abstract Theory of Bulking ⋆

M. Delorme [a], J. Mazoyer [a], N. Ollinger [b,1], G. Theyssier [c]

[a] *LIP, ENS Lyon, CNRS, 46 allée d'Italie, 69 007 Lyon, France*

[b] *LIF, Aix-Marseille Université, CNRS, 39 rue Joliot-Curie, 13 013 Marseille, France*

[c] *LAMA, Université de Savoie, CNRS, 73 376 Le Bourget-du-Lac Cedex, France*

Abstract

This paper is the first part of a serie of two papers dealing with bulking: a quasi-order on cellular automata comparing space-time diagrams up to some rescaling. Bulking is a generalization of grouping taking into account universality phenomena, giving rise to a maximal equivalence class. In the present paper, we discuss the proper components of grouping and study the most general extensions. We identify the most general space-time transforms and give an axiomatization of bulking quasi-order. Finally, we study some properties of intrinsically universal cellular automata obtained by comparing grouping to bulking.

Key words: cellular automata, bulking, grouping, classification

First studies of cellular automata, by von Neumann [14] and others [3], were concerned with the following question. How to construct, using only local interactions, some object with a given global behavior? In the 50s, the basic idea was to cut the global behavior into smaller components, still global, and to describe interactions between these new components. Thus, 2D Turing-universal cellular automata of that time were described in terms of tape, code, arm and head. In this setting, the main difficulty was to describe each component using local interactions in such a way that it will be possible to merge all the needed interactions of the different components. In a second time, the question of optimization (small number of states) has appeared. As it is important to point out, the description of global component and their interactions was given not formally, in a mathematical language, but using analogies with the real world.

Later, as the problematic of constructing global behavior with local interactions was still studied (*French Flag*, *Firing Squad*, ...), some more mathe-

⋆ This work was supported by French ANR project Sycomore

[1] Corresponding author (Nicolas.Ollinger@lif.univ-mrs.fr)

matical works began studying the set of cellular automata, first focusing on properties like the one-to-one character of 1D cellular automata. These studies put into light an ambiguity: one might choose either to fix the neighborhood or the number of states, see Hedlund [6]. In the 60s, the question of optimization of the number of states of a self-reproducing or universal cellular automata was studied.

Simultaneously, in the 60s, the methods of constructions of automata have evolved. People like Fischer [5] but also people studying *French Flag* and *Firing Squad* have understood that, for the major part, the algorithmic of construction of automata was geometric involving continuous lines (straight lines, parabolas, exponentials) seen as carrying information and that the algorithm, very often, was mainly to choose atoms of information to combine, to organize their birth, their meetings and their death defining the "continuous" lines on which these atoms will move. Then, it was, and it is always, required to translate this arrangement of global components into local interactions. At this point, a choice about the scale between the continuous description and the discrete implementation appears: for example, is a straight line thickness represented by a state or by a pattern? is the discretization scale the same for space and time? These choices give automata with an identical global behavior but with different number of states or different neighborhood. The question is, how do one translate one of these cellular automata into the others to identify the common behavior?

In modern words, the question is: given a matrix $\mathbb{Z} \times \mathbb{N}$ of letters in S, known to be an evolution of some cellular automaton, how can we tile $\mathbb{Z} \times \mathbb{N}$ in such a way that patterns belonging to tiles may be considered as states of another cellular automaton? The number of tiles of the tiling to be defined is finite because we aim to obtain a cellular automaton. We observe that the choice of a matrix $\mathbb{Z} \times \mathbb{N}$, as opposed to $\mathbb{N} \times \mathbb{N}$, forbids to propagate some information concerning the index of the initial configuration (like *this is cell* 0). Therefore, the shape of tiles are not allowed to depend on the states they contain. To find a tiling independent of the automaton implies that all the tiles are the same. Without this condition of uniformity allows more tilings, see Róka [13], Martin II [10]. We study this problematic with the additional assumption that the tile, tiling $\mathbb{Z}\times\mathbb{N}$, is connected. The main result is that the only possible tiles are the "natural" ones; that is shifted rectangles. Doing so, we have identified classes of automata which have the same global properties up to a change of scale of the space-time diagrams.

In some way, we are now able to compare global properties. We say that a class of cellular automata simulates another one if all orbits of all automata in the simulated class are orbits of subautomton of some automaton of the simulating class. In this case, all global properties appearing in the simulated class appear in the simulating class. This simulation relation induces an order on classes

of automata. This order depends on the shapes of tiles allowed: we consider all possible connected tiles. The restriction to square of tiles corresponds to grouping [12]. In these two cases, the order has trivially a minimum (automata with only one state). In the general case, a maximum class exists but not in the restricted family. An automaton belonging to the maximum class is called *intrinsically universal* because it is able to simulate, in the defined sense, all automata. This notion of intrinsic universally is, in fact, very old: it may be found in Banks [2], Albert and Čulik [1]; a first attempt of formalization can be found in Martin [9]. Since twenty years, people have observed that if a state of an initial configuration of some automaton $\mathcal{A}$ is coded, in the previously defined way, as ℓ states in an initial configuration of an intrinsically universal automaton $\mathcal{B}$, to obtain the following configuration of $\mathcal{A}$ needs strictly more than ℓ transitions of $\mathcal{B}$. The fact that the family of simulations by squares has no maximum class proves this observation. In some way, intrinsically universally implies a time for information movement and a time for computation that cannot be parallelized. Finally, intrinsically universally is different and more powerful than Turing universality (see [4]).

In section 1, definitions are given with a geometrical bias. In section 2, grouping is presented. In section 3, possible extensions are investigated, searching for good candidates of geometrical transforms and elementary simulation relation. In section 4, bulking is defined as a formal family of simulation quasi-orders and an extension of grouping is chosen. In section 5, this extension is compared to grouping and a first result concerning intrinsically universal cellular automata is obtained.

1 Patterns, Colorings and Cellular Automata

1.1 Patterns

A *pattern* $\mathcal{P}$ is a subset of $\mathbb{Z}^d$. The *m-rectangular pattern* $\boxplus_m$ is the pattern $\{0,\dots,m_1-1\}\times\cdots\times\{0,\dots,m_d-1\}$. Using the natural extension of $+$ on sets, the *translation* $\mathcal{P}+u$ of $\mathcal{P}$ by a vector $u\in\mathbb{Z}^d$ is the pattern $\{z+u|z\in\mathcal{P}\}$ and the *sum* of two patterns $\mathcal{P}$ and $\mathcal{P}'$ is the pattern $\{z+z'|z\in\mathcal{P},z'\in\mathcal{P}'\}$. An *elementary translation* is a translation by a vector $\varsigma_k\in\mathbb{Z}^d$ with all coordinates equal to 0 but the kth which is equal to 1 or -1. Every translation is obtained by composition of elementary translations.

A *basis* $V\in\left(\mathbb{Z}^d\right)^d$ is a tuple of d non-zero linearly independent vectors $(v_1,\dots,v_d)$. The *m-rectangular basis* $\square_m$ is the basis $(m_1\delta_1,\dots,m_d\delta_d)$ where δ_k has all coordinates equal to 0 but the kth which is equal to 1. The *image* $V\odot z$ by V of a point $z\in\mathbb{Z}^d$ is the point $\sum_{i=1}^d z_iv_i$. The *image* $V\odot\mathcal{P}$ by V of

a pattern $\mathcal{P}$ is the pattern $\{V \odot z | z \in \mathcal{P}\}$. The *image* by a basis V' of a basis V is the basis $V' \odot V = (V' \odot v_1, \dots, V' \odot v_d)$. Notice that $\square_m \odot \square_{m'} = \square_{mm'}$ where for all k, $(mm')_k = m_k m'_k$.

A *tiling of space* is a pair $(\mathcal{P}, V)$ where $\mathcal{P}$ is a pattern that tiles the plane with the basis V, *i.e.* such that $\left\{\mathcal{P} + V \odot z \middle| z \in \mathbb{Z}^d\right\}$ is a partition of $\mathbb{Z}^d$. The *m-rectangular tiling* is the tiling $(\boxplus_m, \square_m)$. The *composition* $(\mathcal{P}', V') \circ (\mathcal{P}', V')$ of two tilings of space $(\mathcal{P}', V')$ and $(\mathcal{P}, V)$ is the tiling of space $(\mathcal{P} + V \odot \mathcal{P}', V' \odot V)$. Notice that $(\boxplus_m, \square_m) \circ (\boxplus_{m'}, \square_{m'}) = (\boxplus_{mm'}, \square_{mm'})$. Notice that the size of $\mathcal{P}$ has to be $|\det V|$. Given a basis V, the equivalence relation $\equiv_V$ on $\mathbb{Z}^d$ is defined by $z \equiv_V z'$ if $z' - z \in V \odot \mathbb{Z}^d$. It defines precisely $|\det V|$ equivalence classes. Valid patterns are precisely patterns consisting of one point in each equivalence class of $\equiv_V$.

1.2 Colorings

A *coloring* $\mathcal{C} \in \Sigma^{\mathcal{P}}$ is a covering of its support $\mathcal{P} \subseteq \mathbb{Z}^d$, denoted as $\mathrm{Sup}(\mathcal{C})$, by letters of a finite alphabet Σ. A *singleton coloring* is a coloring with a singleton support. A *finite coloring* is a coloring with finite support. A *full coloring* is a coloring with support $\mathbb{Z}^d$. A coloring $\mathcal{C}'$ is a *subcoloring* of a coloring $\mathcal{C}$, denoted as $\mathcal{C}' \ll \mathcal{C}$, if $\mathcal{C}'$ is a restriction of $\mathcal{C}$, *i.e.* $\mathcal{C}' = \mathcal{C}_{|\mathrm{Sup}(\mathcal{C}')}$. The *translation* $u \cdot \mathcal{C}$ of $\mathcal{C}$ by a vector $u \in \mathbb{Z}^d$ is the coloring with support $\mathrm{Sup}(\mathcal{C}) + u$ satisfying, for all $z \in \mathrm{Sup}(\mathcal{C})$, $u \cdot \mathcal{C}(z + u) = \mathcal{C}(z)$. The *u-shift* is the translation map over full colorings $\sigma_u : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ defined for all coloring $\mathcal{C}$ by $\sigma_u(\mathcal{C}) = u \cdot \mathcal{C}$. An *elementary shift* is a shift by an elementary translation. A coloring $\mathcal{C}$ occurs in a coloring $\mathcal{C}'$, denoted as $\mathcal{C} \Subset \mathcal{C}'$ if a translation of $\mathcal{C}$ is a subcoloring of $\mathcal{C}'$. A coloring $\mathcal{C}$ is *periodic*, with periodicity vector $u \in \mathbb{Z}^d$, if for all $z \in \mathrm{Sup}(\mathcal{C}) \cap (\mathrm{Sup}(\mathcal{C}) - u)$, $\mathcal{C}(z) = \mathcal{C}(z + u)$. A coloring $\mathcal{C}$ is *s-finite* if $\mathcal{C}$ is equal to s everywhere but on a finite support.

The *cylinder* generated by a coloring $\mathcal{C}$ over an alphabet Σ is the set of full colorings $[\mathcal{C}] = \left\{\mathcal{C}' \in \Sigma^{\mathbb{Z}^d} \middle| \mathcal{C} \ll \mathcal{C}'\right\}$. The *Cantor topology* on $\Sigma^{\mathbb{Z}^d}$ is the product topology of the discrete topology on Σ. Its open sets are generated by the cylinders of finite colorings. This topology is compact, metric and perfect.

The *packing map* with tiling $(\mathcal{P}, V)$ over the alphabet Σ is the map $\langle \mathcal{P}, V \rangle : \Sigma^{\mathbb{Z}^d} \to (\Sigma^{\mathcal{P}})^{\mathbb{Z}^d}$ defined for all full coloring $\mathcal{C} \in \Sigma^{\mathbb{Z}^d}$ and all point $z \in \mathbb{Z}^d$ by $\langle \mathcal{P}, V \rangle (\mathcal{C})(z) = ((-V \odot z) \cdot \mathcal{C})_{|\mathcal{P}}$. The *rectangular packing map* $\square^m$ is the packing map $\langle \boxplus_m, \square_m \rangle$, its inverse is denoted as $\square^{-m}$.

1.3 Cellular Automata

A *d-dimensional cellular automaton* (d-CA) $\mathcal{A}$ is a triple (S, N, f) where S is a finite set of states, N is the neighborhood, a finite pattern of $\mathbb{Z}^d$ and $f : S^N \to S$ is the local rule of $\mathcal{A}$. A *configuration* of $\mathcal{A}$ is a mapping $c \in S^{\mathbb{Z}^d}$. The *global transition function* $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ of $\mathcal{A}$ maps every configuration $c \in S^{\mathbb{Z}^d}$ to the configuration $G(c) \in S^{\mathbb{Z}^d}$ such that, for all $z \in \mathbb{Z}^d$, $G(c)(z) = f((-z \cdot c)_{|N})$. The *space-time diagram* of $\mathcal{A}$ starting from a configuration c_0 is a mapping $\Delta \in S^{\mathbb{N}\times\mathbb{Z}^d}$ encoding an infinite sequence of successive orbits of the dynamical system $\left(S^{\mathbb{Z}^d}, G\right)$ by $\Delta(0) = c_0$ and, for all $t \in \mathbb{Z}^+$, $\Delta(t) = G(\Delta(t-1))$. The set of space-time diagrams of a CA $\mathcal{A}$ is denoted as $\mathrm{Diag}\,\mathcal{A}$. A d-CA is *autarkic* if its neighborhood is $\{0\}$. Every d-CA with a singleton neighborhood is the composition of a shift by an autarkic CA.

A d-CA $\mathcal{A}$ is a *subautomaton* of a d-CA $\mathcal{B}$, with respect to the injective map $\varphi : S_\mathcal{A} \to S_\mathcal{B}$, denoted as $\mathcal{A} \sqsubseteq_\varphi \mathcal{B}$, if $G_\mathcal{B} \circ \overline{\varphi} = \overline{\varphi} \circ G_\mathcal{A}$ where $\overline{\varphi} : S_\mathcal{A}^{\mathbb{Z}^d} \to S_\mathcal{B}^{\mathbb{Z}^d}$ is the canonical extension of φ defined for all $c \in S^{\mathbb{Z}^d}$ by $\overline{\varphi}(c) = \varphi \circ c$. Equivalently stated, a d-CA $\mathcal{A}$ is a subautomaton of a d-CA $\mathcal{B}$ with respect to φ if and only if $\overline{\varphi}\,(\mathrm{Diag}\,\mathcal{A}) \subseteq \mathrm{Diag}\,\mathcal{B}$. A d-CA $\mathcal{A}$ is *isomorphic* to a d-CA $\mathcal{B}$, denoted as $\mathcal{A} \equiv \mathcal{B}$, if both $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\mathcal{B} \sqsubseteq \mathcal{A}$.

In this paper, we focus on CA seen as discrete dynamical systems, that is the pair $(S^{\mathbb{Z}^d}, G)$ up to isomorphism and more precisely the orbits of such systems (represented by space-time diagrams). Thanks to the following theorem, we can freely manipulate global rules of CA to generate new CA.

Theorem 1 (Hedlund [7]) *A map $G : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ is the global transition function of a cellular automaton if and only if G is continuous and commutes with elementary translations.*

PROOF. The preimage by a continuous map $G : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ of a cylinder generated by a singleton coloring is an open set: a finite union of cylinders generated by finite colorings. If G commutes with elementary translations, thus with translations, it is a d-CA. Moreover, the global transition function of a CA is, by construction, continuous and commutes with translations. ■

A CA is injective (*resp.* surjective, bijective) if its global rule is injective (*resp.* surjective, bijective). By previous theorem, the composition of two CAs, the cartesian product of two CAs or the inverse of a bijective CA is a CA. The *cartesian product* of two d-CA $\mathcal{A}$ and $\mathcal{B}$ is the d-CA $\mathcal{A} \times \mathcal{B}$ whose global transition function verifies for all $(c, c') \in S_\mathcal{A}^{\mathbb{Z}^d} \times S_\mathcal{B}^{\mathbb{Z}^d}$, $G_{\mathcal{A}\times\mathcal{B}}((c, c')) = (G_\mathcal{A}(c), G_\mathcal{B}(c'))$.

The *phase space* of a CA $(S^{\mathbb{Z}^d}, G)$ is the graph with vertices $S^{\mathbb{Z}^d}$ and two kinds

of directed edges: global rule edges are pairs $(c, G(c))$ labelled by G, translation edges are pairs $(c, \varsigma_i \cdot c)$ labelled by ς_i, for all $c \in S^{\mathbb{Z}^d}$ and elementary translation ς_i. Orbits correspond to infinite G-paths in the phase space. A *periodic point*, with period $p \in \mathbb{Z}^+$, is a configuration c such that $G^p(c) = c$. A *fixpoint* is a periodic point with period 1. A *Garden-of-Eden* is a configuration c with no ancestor, *i.e.* such that $G^{-1}(c) = \emptyset$. An *ultimately periodic point*, with transient $\tau \in \mathbb{N}$ and period $p \in \mathbb{Z}^+$, is a configuration c such that $G^{p+\tau}(c) = G^\tau(c)$.

The *limit set* Λ_G of a CA $(S^{\mathbb{Z}^d}, G)$ is the non-empty translation invariant compact set $\Lambda_G = \bigcap_{i\in\mathbb{N}} \Lambda_G^{(i)}$ where $\Lambda_G^{(0)} = S^{\mathbb{Z}^d}$ and for all $i \in \mathbb{N}$, $\Lambda_G^{(i+1)} = G\left(\Lambda_G^{(i)}\right)$. The limit set consists exactly of all configurations that appear in biinfinite space-time diagrams $\Delta \in S^{\mathbb{Z}\times\mathbb{Z}^d}$ such that, for all $t \in \mathbb{Z}$, $\Delta(t+1) = G(\Delta(t))$. A CA is *nilpotent* if its limit set is a singleton. By compacity, a CA $(S^{\mathbb{Z}^d}, G)$ is nilpotent if and only if there exists a uniform bound $\tau \in \mathbb{Z}^+$ such that $G^\tau(S^{\mathbb{Z}^d})$ is a singleton.

A *d-dimensional partitioned cellular automaton* (d-PCA) $\mathcal{A}$ is a triple (S, N, ψ) where S is a finite set of states, N is the neighborhood, a finite pattern of $\mathbb{Z}^d$ and $\psi : S^N \to S^N$ is the local rule of $\mathcal{A}$. The N-mixing rule $\mu_N : (S^N)^{\mathbb{Z}^d} \to (S^N)^{\mathbb{Z}^d}$ is defined, for all $c \in (S^N)^{\mathbb{Z}^d}$, for all $z \in \mathbb{Z}^d$ and for all $u \in N$, by $\mu_N(c)(z)(u) = c(z+u)(u)$. The global transition function of $\mathcal{A}$ is $\overline{\psi} \circ \mu_N$. Every PCA is a CA. Moreover, (the global transition function of) a PCA is bijective if and only if its local rule is bijective.

2 Grouping Cellular Automata

The grouping quasi-order was introduced by Mazoyer and Rapaport [12] as a successful tool to classify CA according to algebraic properties [11]. However, grouping fails to capture several geometrical properties of CA that one would like to see classified by such a geometric classification. In this section, we recall the grouping quasi-order.

Grouping deals with cellular automata of dimension 1 and neighborhood $\{-1, 0, 1\}$. Such a cellular automaton will be denoted in this section as a pair (S, f). In space-time diagrams of such cellular automata, a k-uple of state of a segment of k cells at time t only depends on states of $2t + k$ states at time 0, as shown on Fig. 1.

The nth iteration of the local rule is recursively defined by $f^1 = f$ and, for all

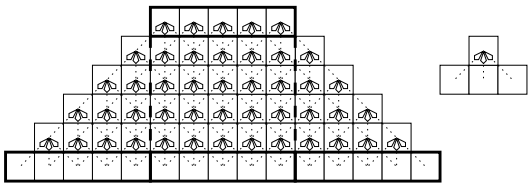


Fig. 1. dependencies in space-time diagrams for grouping

$n \in \mathbb{N}$,

$$f^{n+1}(x_{-n-1},\dots,x_{n+1}) = f(f^n(x_{-n-1},\dots,x_{n-1}),\\ f^n(x_{-n},\dots,x_{n}),\\ f^n(x_{-n+1},\dots,x_{n+1})) \quad .$$

Inspired by dependencies in space-time diagrams and geometrical considerations, one defines the nth grouped instance $(S,f)^{[n]}$ of a cellular automaton (S,f) by $(S,f)^n = (S^n, f^n_\square)$ where $f^n_\square$ is defined for all triple of n-uple of states by

$$f^n_\square((x_1,\dots,x_n),(x_{n+1},\dots,x_{2n}),(x_{2n+1},\dots,x_{3n})) = \big(f^n(x_1,\dots,x_{2n+1}),\\ f^n(x_2,\dots,x_{2n+2}),\\ \vdots\\ f^n(x_n,\dots,x_{3n})\big) \quad .$$

For all $n > 0$, the space-time diagrams of a cellular automaton (S,f) are in one-to-one correspondance with the space-time diagrams of its nth grouped instance $(S,f)^{[n]}$.

A cellular automaton (S,f) is *simulated* by a cellular automaton (S',f'), denoted by $(S,f) \leqslant_\square (S',f')$, if there exists two powers m and n such that $(S,f)^m \sqsubseteq (S',f')^n$.

Theorem 2 ([12]) *The relation $\leqslant_\square$ is a quasi-order relation.*

Theorem 3 ([12]) *$\leqslant_\square$ admits no maximal element.*

3 Towards a Generalization of Grouping

Several extensions of grouping are possible: change the elementary simulation order (subautomaton), several extensions of the allowed geometrical transformations. Subsection 3.1 points out the interest of the subautomaton relation

to obtain all CA as an algebraic closure, subsection 3.2 shows connections between grouping and stability of a ertain kind of subshifts and the need of new geometrical transformations to that extent, subsection 3.3 characterizes the most general family of space-time transformations preserving CA uniformity.

3.1 An Algebraic Characterization of Cellular Automata

Every CA is the subautomaton of a PCA. The mixing part of a PCA rule is a cartesian product of shifts, that is composition of elementary shifts; the local rule of a PCA acts as an autarkik CA. When restricting PCA to RPCA, all RCA are generated. From there, one derives the following algebraic characterizations of CA and RCA, pointing out the use of the subautomaton relation to hide blueprint marks.

Theorem 4 *The set of d-CA is the algebraic closure of autarkic CA and elementary shifts by composition, cartesian product and subautomaton.*

PROOF. Autarkic CA and shifts being CA, the closure generates only CA.

Let (S, N, f) be a d-CA $\mathcal{A}$. Let $\varphi : S \to S^N$ map s to $(s, \dots, s)$. By construction, $\mathcal{A}$ is a sub-automaton of the PCA $(S, N, \varphi \circ f)$ with respect to φ. The global transition function of the PCA is the composition of $\overline{\varphi \circ f}$, which is an autarkic CA, by μ_N. The N-mixing map μ_N is the product of $|N|$ shifts, each of which can be obtained as a composition of elementary shifts. ■

The restriction to RCA uses the following fact: every RCA is a subautomaton of a RPCA with neighborhood a valid neighborhood for both the RCA and its reverse. As reversibility is undecidable starting from dimension 2, there exists RCA with arbitrarily larger RPCA representation than PCA representation.

Theorem 5 *The set of d-RCA is the algebraic closure of bijective autarkic CA and elementary shifts by composition, cartesian product and subautomaton.*

PROOF. Injectivity being preserved by composition, cartesian product and subautomaton, the closure generates only RCA.

Let (S, N, f) be a d-RCA $\mathcal{A}$ with its reverse (S, N, g) a d-RCA $\mathcal{B}$. Let $S_\bullet = S \cup \{\bullet\}$. To conclude, we introduce three RPCA $\mathcal{A}^\bullet = (S_\bullet^2, N, f_\bullet)$, $\mathcal{B}^\bullet = (S_\bullet^2, -N, g_\bullet)$ and $\mathcal{S} = (S_\bullet^2, N, h)$ such that $\mathcal{A} \sqsubseteq_\varphi \mathcal{S} \circ \mathcal{B}^\bullet \circ \mathcal{A}^\bullet$ where φ maps s to $((s, \bullet), \dots, (s, \bullet))$.

The bijective map $f_\bullet : S_\bullet^2 \to S_\bullet^2$ is given by the following partial injective definition. For all $(s_1, \dots, s_k) \in S^N$, let $f_\bullet((s_1, \bullet), \dots, (s_k, \bullet)) = ((s_1, s'), \dots, (s_k, s'))$ where $s' = f(s_1, \dots, s_k)$.

The bijective map $g_\bullet : S_\bullet^2 \to S_\bullet^2$ is given by the following partial injective definition. For all $(s_1, \dots, s_k) \in S^N$, let $g_\bullet((s', s_1), \dots, (s', s_k)) = ((\bullet, s_1), \dots, (\bullet, s_k))$ where $s' = g(s_1, \dots, s_k)$.

The bijective map $h : S_\bullet^2 \to S_\bullet^2$ is given by the following partial injective definition. For all $(s_1, \dots, s_k) \in S^N$, let $h((\bullet, s_1), \dots, (\bullet, s_k)) = ((s_1, \bullet), \dots, (s_k, \bullet))$.

Using the arguments of the proof of Thm 4, the global transition function of every RPCA is expressible in the closure. ■

3.2 Grouping and stability of bloc subshifts

Note: for clarity and within this subsection only, we restrict to dimension 1.

A subautomaton of a given CA $\mathcal{A}$ is always induced by a subset of states which is stable under iterations, *i.e.* a set $T \subseteq S_\mathcal{A}$ such that $G_\mathcal{A}\Big(T^\mathbb{Z}\Big) \subseteq T^\mathbb{Z}$. Sets of the form $T^\mathbb{Z}$ are called *full-shifts* which are particular *subshifts* (closed translation-invariant set of configurations).

We can establish a similar connection between the grouping relation $\leqslant_\square$ and a particular kind of subshifts that we call *bloc subshifts*. A bloc subshift is the set of configuration obtained by (infinite) catenation of finite words of same length from a given set. Formally, given an integer m and a set X of words of length m, the bloc subshift Σ_X associated to X is the set of configurations whose language is X^* closed by subword (a subshift is characterised by the language of its configurations, see [8]). If $G_\mathcal{A} \sqsubseteq_\phi G_\mathcal{B}^{[i]}$, then $X = \phi(S_\mathcal{A})$ is a set of words of length i over the alphabet $S_\mathcal{B}$. It is straightforward to check that the bloc subshift Σ_X is (weakly) stable under the action of $G_\mathcal{B}$, more precisely:

$$G_\mathcal{B}^i(\Sigma_X) \subseteq \Sigma_X.$$

Therefore any subautomaton with q states of a grouped instance of $\mathcal{B}$ is induced by a bloc subshift made from q words which is (weakly) stable under the action $\mathcal{B}$. The converse is false as shown by the following example: a CA can have a weakly stable bloc subshift made from q words without any subautomaton with q states in the corresponding grouped instance.

Example 6 *Consider $\mathcal{A}$ over state set $S_\mathcal{A} = \{0,1\} \times \{0,1\}$ with neighbourhood $\{0,1\}$ and local rule f defined by*

$$f\Big((a_1, b_1), (a_2, b_2)\Big) = (b_1, a_2).$$

$G^2_{\mathcal{A}}$ is the elementary right-shift CA over state set $S_{\mathcal{A}}$. So for any set X of words of length 2 over alphabet $S_{\mathcal{A}}$, the bloc subshift Σ_X is stable under $G^2_{\mathcal{A}}$. Now consider ${G_{\mathcal{A}}}^{[2]}$. The only stable subset of states it admits are of the form $Q \times Q \subseteq S_{\mathcal{A}} \times S_{\mathcal{A}}$ since $G^2_{\mathcal{A}}$ is an elementary shift. Therefore, a subautomaton of ${G_{\mathcal{A}}}^{[2]}$ must have a square number of states. ■

However, as shown by the following theorem, a larger set of geometrical transformations allows to capture all weakly stable bloc subshifts. This constitutes an additional motivation for the generalisation of grouping presented in the sequel.

Theorem 7 *Let i, m, q be positive integers, and $\mathcal{B}$ be a CA. The two following propositions are equivalent:*

(1) there exists a set X of q words of length m such that $G^i_{\mathcal{B}}\Big(\Sigma_X\Big) \subseteq \Sigma_X$;
(2) there exists a translation s and a CA $\mathcal{A}$ with q states which is a subautomaton of $\square^m \circ s \circ G^i_{\mathcal{B}} \circ \square^{-m}$

PROOF. First, for (2) $\Rightarrow$ (1), we suppose $G_{\mathcal{A}} \sqsubseteq_\phi \square^m \circ s \circ G^i_{\mathcal{B}} \circ \square^{-m}$ and it suffices to check that $X = \phi(S_{\mathcal{A}})$ is a set of q words of length m and that $G^i_{\mathcal{B}}\Big(\Sigma_X\Big) \subseteq \Sigma_X$ (by definition of $\sqsubseteq$ and by commutation of $G_{\mathcal{B}}$ with translations).

For (1) $\Rightarrow$ (2), we suppose 1 and consider the set E_p of configurations from Σ_X for which the catenation of words of X is aligned with position p of the lattice, formally:

$$E_p = \Big\{c \in \Sigma_X : \forall k \in \mathbb{Z}, c(km+p)\cdots c(km+p+k-1) \in X\Big\}.$$

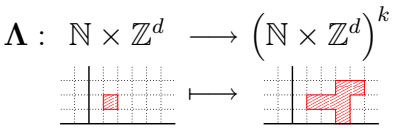
Clearly, E_p are closed sets and $\Sigma_X = \cup_p E_p$. Moreover, $\square^m(E_0)$ is a full-shift of alphabet X. Now consider a configuration $c \in \square^m(E_0)$ whose language is X^* (a "universal" configuration as called sometimes in the literature) and let $c' = \square^{-m}(c)$. By hypothesis, $G^i_{\mathcal{B}}(c') \in \Sigma_X$ so it belongs to some E_p. By choice of c', any $c'' \in E_0$ is obtained as the limit of some sequence $\Big(t_n(c')\Big)_n$ where each t_n is a translation of a vector multiple of m. By continuity and commutation with translations of $G_{\mathcal{B}}$ we deduce that $G_{\mathcal{B}}(c'')$ is the limit of elements of E_p so it belongs to E_p because this set is closed. Hence, we have $G^i_{\mathcal{B}}\Big(E_0\Big) \subseteq E_p$ and $s \circ G^i_{\mathcal{B}}\Big(E_0\Big) \subseteq E_0$ if s is the suitable translation. From this we deduce that $\square^m(E_0)$ is a non-trivial stable full-shift of $\square^m \circ s \circ G^i_{\mathcal{B}} \circ \square^{-m}$ and the theorem follows by the discussion at the beginning of this section. ■

3.3 A Characterization of the Most General Geometrical Space-Time Transforms

3.3.1 Geometrical Space-Time Transforms

Grouping and classical transforms described in previous section consists of purely geometrical transforms: transforms that do not depend on the state set of the transformed CA, and thus can be applied to all CA. Such a transform maps a space-time diagram to a new space-time diagram, each space-time cell of which consists of a tuple of space-time cells of the initial diagram.

Formally, a *geometrical transform* is a pair $(k, \mathbf{\Lambda})$ where $k \in \mathbb{Z}^+$ and $\mathbf{\Lambda}$ maps $\mathbb{N} \times \mathbb{Z}^d$ to $\left(\mathbb{N} \times \mathbb{Z}^d\right)^k$. To help the read visualize the transform, we will depict geometrical transforms as follows:



The *space-time diagram transform* over a state set S by a geometrical transform $(k, \mathbf{\Lambda})$, is the map $\overline{\mathbf{\Lambda}}_S$ from $S^{\mathbb{N}\times\mathbb{Z}^d}$ to $S^{\left(\mathbb{N}\times\mathbb{Z}^d\right)^k}$ defined, for all space-time diagram $\Delta \in S^{\mathbb{N}\times\mathbb{Z}^d}$ and for all space-time position $\xi \in \mathbb{N} \times \mathbb{Z}^d$, by $\overline{\mathbf{\Lambda}}_S(\xi) = (\Delta(\lambda_1), \ldots, (\Delta(\lambda_k))$ where $\mathbf{\Lambda}(\xi) = (\lambda_1, \ldots, \lambda_k)$.

Example 8 *In 1D, the geometrical transform* $(3, \mathbf{\Lambda}^{\langle 3,4,1\rangle})$ *defined, for all* $(t, p) \in \mathbb{N} \times \mathbb{Z}$ *by* $\mathbf{\Lambda}^{\langle 3,4,1\rangle}(t, p) = ((4t, 3p + t), (4t, 3p + t + 1), (4t, 3p + t + 2))$ *transforms the set of space-time diagrams* $\operatorname{Diag} \mathcal{A}$ *of every CA* $\mathcal{A}$ *into the set of space-time diagrams* $\operatorname{Diag} \mathcal{A}^{\langle 3,4,1\rangle}$ *of a new CA* $\mathcal{A}^{\langle 3,4,1\rangle}$*. Fig. 2 depicts the transform.* ◇

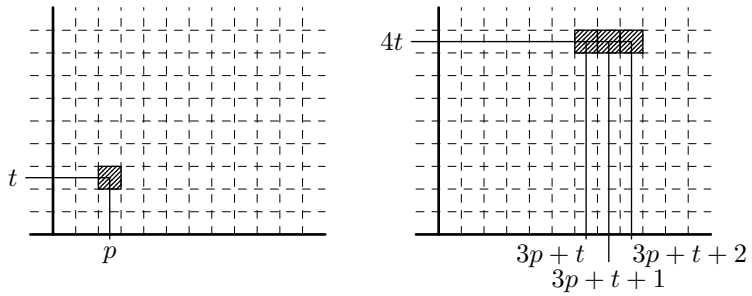


Fig. 2. Sample geometrical transform: $\left(3, \mathbf{\Lambda}^{\langle 3,4,1\rangle}\right)$

The *composition* $(k', \mathbf{\Lambda}') \circ (k, \mathbf{\Lambda})$ *of two geometrical transforms* $(k, \mathbf{\Lambda})$ and

$(k', \mathbf{\Lambda}')$ is the geometrical transform $(kk', \mathbf{\Lambda}' \circ \mathbf{\Lambda})$ where, for all $\xi \in \mathbb{N} \times \mathbb{Z}^d$:

$$\left(\mathbf{\Lambda}' \circ \mathbf{\Lambda}\right)(\xi) = \left(\mathbf{\Lambda}\left(\mathbf{\Lambda}'(\xi)_1\right)_1, \ldots, \mathbf{\Lambda}\left(\mathbf{\Lambda}'(\xi)_{k'}\right)_k\right) \quad .$$

Let $\widetilde{\mathbf{\Lambda}}$ map each space-time cell to the set of associated space-time cells by $\mathbf{\Lambda}$:

$$\begin{aligned} \widetilde{\mathbf{\Lambda}} : 2^{\mathbb{N}\times\mathbb{Z}^d} &\longrightarrow 2^{\mathbb{N}\times\mathbb{Z}^d} \\ X &\longmapsto \bigcup_{\xi\in X} \{\mathbf{\Lambda}(\xi)_1, \ldots, \mathbf{\Lambda}(\xi)_k\} \end{aligned}$$

A *nice geometrical transform* is a geometrical transform which plays nicely with space-time diagrams and can be used to extend grouping. It should transform sets of space-time diagrams into sets of space-time diagrams and be non-trivial (to avoid some cheating).

Formally, a geometrical transform $(k, \mathbf{\Lambda})$ is *nice* if it satisfies the following conditions:

(i) for all CA $\mathcal{A}$, there exists a CA $\mathcal{B}$ such that $\overline{\mathbf{\Lambda}}_{S_\mathcal{A}}(\mathrm{Diag}\,\mathcal{A}) = \mathrm{Diag}\,(B)$
(ii) for all time $t \in \mathbb{N}$, $\widetilde{\mathbf{\Lambda}}(\{t+1\} \times \mathbb{Z}^d) \not\subseteq \widetilde{\mathbf{\Lambda}}(\{t\} \times \mathbb{Z}^d)$.

Nice geometrical transforms are close by composition.

3.3.2 Packing, Cutting and Shifting

The classical transforms of previous section can be expressed as composition of three kinds of nice transforms, the action of which can be expressed easily in an algebraic way as global rules compositions.

Packing. A purely spatial geometrical transform can be defined using a tiling of space to cut space regularly. The *packing transform* $\mathbf{P}_{\mathcal{P},V}$, with tiling of space $(\mathcal{P}, V)$, is defined for all $(t, p) \in \mathbb{N} \times \mathbb{Z}^d$ by:

$$\mathbf{P}_{\mathcal{P},V}(t,p) \;=\; \{t\} \times (\mathcal{P} + V \odot p) \quad .$$

Let $\mathcal{A}$ be a CA and $(\mathcal{P}, V)$ a tiling of space, the image of $\mathrm{Diag}\,\mathcal{A}$ by $\mathbf{P}_{\mathcal{P},V}$ is the set of space-time diagrams of the CA with global rule $\langle \mathcal{P}, V\rangle \circ G_\mathcal{A} \circ \langle \mathcal{P}, V\rangle^{-1}$.

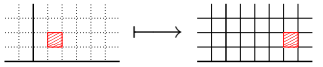
Cutting. A purely temporal geometrical transform can be defined by cutting unwanted time steps. The *cutting transform* $\mathbf{C}_T$, with rate $T \in \mathbb{Z}^+$, is defined

for all $(t,p) \in \mathbb{N} \times \mathbb{Z}^d$ by:

$$\mathsf{C}_T\,(t,p) \;=\; (tT,p)$$

Let $\mathcal{A}$ be a CA and $T \in \mathbb{Z}^+$ a rate, the image of Diag $\mathcal{A}$ by C_T is the set of space-time diagrams of the CA with global rule $G_{\mathcal{A}}^T$.

Shifting. A pure translation geometrical transform can be defined by shifting space-time. The *shifting transform* S_s, with translation vector $s \in \mathbb{Z}^d$, is defined for all $(t,p) \in \mathbb{N} \times \mathbb{Z}^d$ by:



$$\mathsf{S}_s\,(t,p) \;=\; (t,p+ts)$$

Let $\mathcal{A}$ be a CA and $s \in \mathbb{Z}^d$ a translation vector, the image of Diag $\mathcal{A}$ by S_s is the set of space-time diagrams of the CA with global rule $\sigma_s \circ G_{\mathcal{A}}$.

Composition. Let $(\mathcal{P},V)$ be a tiling of space, s be a translation vector and T be a rate, the nice geometrical transform $\mathsf{PCS}_{\mathcal{P},V,T,s}$ is the composition $\mathsf{P}_{\mathcal{P},V} \circ \mathsf{S}_s \circ \mathsf{C}_T$. For all $(t,p) \in \mathbb{N} \times \mathbb{Z}^d$ it satisfies:

$$\mathsf{PCS}_{F,v,T,s}\,(t,p) = \{tT\} \times (\mathcal{P} + V \odot v + ts) \quad .$$

Let $\mathcal{A}$ be a CA, the image of Diag $\mathcal{A}$ by $\mathsf{PCS}_{\mathcal{P},V,T,s}$ is the set of space-time diagrams of the CA with global rule

$$\langle \mathcal{P},V\rangle \circ \sigma_s \circ G_{\mathcal{A}}^T \circ \langle \mathcal{P},V\rangle^{-1} \quad .$$

The set of PCS transforms generated by pure P, C and S transforms is closed by composition:

$$\mathsf{PCS}_{\mathcal{P}_2,V_2,T_2,s_2} \circ \mathsf{PCS}_{\mathcal{P}_1,V_1,T_1,s_1} = \mathsf{PCS}_{\mathcal{P}_1+(\mathcal{P}_2\odot V_1),V_2\odot V_1,T_1T_2,(s_2\odot V_1)+T_2s_1} \quad .$$

3.3.3 Characterizing the Most General Transforms

Theorem 9 *Every nice geometrical transform is a* PCS *transform.*

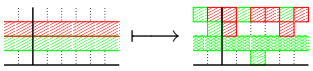
PROOF. Let $(k,\mathbf{\Lambda})$ be a nice geometrical transform. By definition:

(i) for all CA $\mathcal{A}$, there exists a CA $\mathcal{B}$ such that $\overline{\mathbf{\Lambda}}_{S_{\mathcal{A}}}(\text{Diag}\,\mathcal{A}) = \text{Diag}\,(B)$

(ii) for all time $t \in \mathbb{N}$, $\widetilde{\mathbf{\Lambda}}(\{t+1\} \times \mathbb{Z}^d) \not\subseteq \widetilde{\mathbf{\Lambda}}(\{t\} \times \mathbb{Z}^d)$.

The proof proceeds, in 7 steps, by enforcing using (i) successive constraints on $\mathbf{\Lambda}$ until the **PCS** nature becomes clear.

Step 1. Let us first prove the following property:

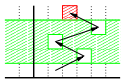


$$\forall t > 0, \forall (t', p_{t'}) \in \widetilde{\mathbf{\Lambda}}\left(\{t\} \times \mathbb{Z}^d\right) \setminus \widetilde{\mathbf{\Lambda}}\left(\{t-1\} \times \mathbb{Z}^d\right), \\ \exists t'' < t', \quad \{t''\} \times \mathbb{Z}^d \subseteq \widetilde{\mathbf{\Lambda}}\left(\{t-1\} \times \mathbb{Z}^d\right)$$

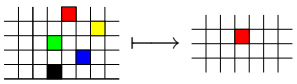
This property states that nice transforms preserves some temporal dependencies: in a transformed space-time diagram, states of cells at time t is completely determined by states of cells at time $t-1$.

Assume that the property is not satisfied at time t. Thus, there exists a time t' and a sequence of spatial positions $(p_0, \ldots, p_{t'})$ such that $(t', p_{t'})$ participates to a transformed cell at time t and for all time i the cell (i, p_i) does not participate to a transformed cell at time $t-1$.

$$\begin{cases} \forall i \leqslant t', & (i, p_i) \notin \widetilde{\mathbf{\Lambda}}\left(\{t-1\} \times \mathbb{Z}^d\right) \\ & (t', p_{t'}) \in \widetilde{\mathbf{\Lambda}}\left(\{t\} \times \mathbb{Z}^d\right) \end{cases}$$



Let $\mathcal{A}$ be a CA with state set $\{\perp, 0, \ldots, t'\}$, neighorhood radius at least $\max_{i,j} |p_j - p_i|$ and such that, starting from an initial configuration uniformly equal to $\perp$ but in position p_0 where it is equal to 0, it generates a space-time diagram Δ whose $t'+1$ first configurations are uniformly equal to $\perp$ but, for each time i, the position p_i is equal to i.



The transformed coloring $\overline{\mathbf{\Lambda}}_{S_{\mathcal{A}}}(\Delta)$ is not the space-time diagram of a CA as the configuration at time $t-1$ is uniform and the configuration at time t is not: a CA cannot break such a symmetry.

Step 2. We now show a property on initial configurations:

$$\widetilde{\mathbf{\Lambda}}\left(\{0\} \times \mathbb{Z}^d\right) = \{0\} \times \mathbb{Z}^d$$

This property states that the initial configuration of a transformed space-time diagram is obtained by a purely spatial transformation of the original initial configuration.

It follows from the fact that the image of Diag $\mathcal{A}$ has to be the whole set of space-time diagram of a CA, that is all initial configuration $\left(S^k_{\mathcal{A}}\right)^{\mathbb{Z}^d}$ should be obtained.

Assume that the property is not satisfied. Let p be a position such that there exists $(t, p') \in \widetilde{\mathbf{\Lambda}}(0, p)$ with $t' > 0$. Let $\mathcal{A}$ be the CA with two states $\{0, 1\}$ and local rule $f_{\mathcal{A}}$ constantly equal to 0. For all space-time diagrams of $\mathcal{A}$, the cell (t, p') has state 0, contradicting (i).

With a similar argument, one show that the images of cells at time 0 are composed of disjoined cells:

$$\forall p, q, \quad \text{card}\left(\widetilde{\mathbf{\Lambda}}(0, p)\right) = k \quad \wedge \quad p \neq q \Rightarrow \widetilde{\mathbf{\Lambda}}(0, p) \cap \widetilde{\mathbf{\Lambda}}(0, q) = \emptyset$$

Step 3. Previous property is extended to every time step:

$$\forall t \in \mathbb{N}, \exists t' \in \mathbb{N}, \quad \widetilde{\mathbf{\Lambda}}\left(\{t\} \times \mathbb{Z}^d\right) = \{t'\} \times \mathbb{Z}^d$$

Let $t \in \mathbb{N}$ be a time step and, by **(step 1)**, let t' satisfy

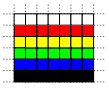
$$\widetilde{\mathbf{\Lambda}}\left(\{t\} \times \mathbb{Z}^d\right) \supseteq \{t'\} \times \mathbb{Z}^d.$$

We show first that each cell at time t contributes a same number l of cells to image cells at time t'. Formally:

$$\begin{aligned} &\exists l, \forall p, \exists i_1 < i_2 \cdots < i_l, \ (\mathbf{\Lambda}(t, p)_{i_1}, \ldots, \mathbf{\Lambda}(t, p)_{i_l}) \in \left(\{t'\} \times \mathbb{Z}^d\right)^l \\ \wedge \quad &\forall i \notin \{i_1, \ldots, i_l\}, \ \mathbf{\Lambda}(t, p)_i \notin \{t'\} \times \mathbb{Z}^d \end{aligned}$$

Let $\mathcal{A}$ be the autarkic CA with state set $\{0, \ldots, t' + 1\}$ whose local rule f satisfies:

$$\forall i, \quad f(i) = \begin{cases} i + 1 & \text{if } i < t' + 1 \\ t' + 1 & \text{if } i = t' + 1 \end{cases}$$



Let Δ be the space-time diagram of $\mathcal{A}$ generated by the uniform configuration with state 0. By **(step 2)**, the initial configuration of $\Delta' = \overline{\mathbf{\Lambda}}_{S_{\mathcal{A}}}(\Delta)$ is also uniform and, by a symmetry argument, every configuration in Δ' is uniform. As a consequence, each cell at time t contains the same number l of component cells in state t'.

Assume that $l < k$. Let $\mathcal{A}$ be the identity CA with state set $\{0, 1\}$. Let $\mathcal{B}$ be the transformed image of $\mathcal{A}$ by $(k, \mathbf{\Lambda})$. Let Δ'_n be the space-time diagram of $\mathcal{B}$ with initial configuration uniformly equal to $(0, \ldots, 0)$ but for a ball of

radius n centered in 0, this ball being filled with state $(1,\dots,1)$. Let Δ_n be the space-time diagram of $\mathcal{A}$ whose image by $\mathbf{\Lambda}$ is Δ'_n. By **(step 2)**, the initial configuration of Δ_n contains exactly kn^d cells with state 1. At time t', the configuration of Δ_n is equal to the configuration at time 0. By **(step 1)**, the configuration of Δ'_n at time t contains at least $\left\lceil \frac{k}{l} n^d \right\rceil$ cells with state different from $(0,\dots,0)$. Thus, the neighborhood radius of $\mathcal{B}$ is at least

$$r_n \geqslant \frac{\left\lceil \frac{k}{l} n^d \right\rceil^{1/d} - n}{t}$$

By hypothesis $\frac{k}{l} > 1$, thus the sequence (r_n) grows unbounded and $\mathcal{B}$ cannot exist.

Step 4. We extend the disjoined block property:

$$\forall t, \forall p, q, \quad \operatorname{card}\left(\widetilde{\mathbf{\Lambda}}(t,p)\right) = k \quad \wedge \quad p \neq q \Rightarrow \widetilde{\mathbf{\Lambda}}(t,p) \cap \widetilde{\mathbf{\Lambda}}(t,q) = \emptyset$$

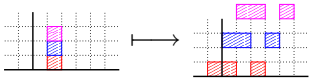
First, we show that $\operatorname{card}\left(\widetilde{\mathbf{\Lambda}}(t,p)\right) = k$: the components of each cell corresponds to disjoined cells. Let $\mathcal{A}$ be the identity CA with state set $\{1,\dots,k\}$. Let Δ be a space-time diagram of $\mathcal{A}$ such that its transformed diagram Δ' has a uniform initial configuration with state $(1,\dots,k)$. By symmetry considerations, the configuration of Δ' at time t is uniformly filled with a state s. By **(step 1)**, s contains all possible component states 1, …, k, thus s is a permutation of $(1,\dots,k)$. Each component corresponds to disjoined cells.

We now show the following property:

$$\forall n, \forall t, \forall p, q, \quad (p \neq q \wedge |p-q| \leqslant n) \Rightarrow \widetilde{\mathbf{\Lambda}}(t,p) \cap \widetilde{\mathbf{\Lambda}}(t,q) = \emptyset$$

This is obtained by generalization of previous symmetry considerations. Let n be a fixed positive integer. Let $\mathcal{A}_n$ be the identity CA with state set $\left\{1,\dots,n^d k\right\}$. Let Δ_n be the space-time diagram of $\mathcal{A}_n$ whose transformed diagram Δ'_n has a periodic initial configuration with periodic coloring the d-dimensional ball with radius n filled with states $(1,\dots,k)$, $(k+1,\dots,2k)$, …, $(n^d-1)k+1,\dots,n^d k)$. By symmetry considerations, the configuration of Δ'_n at time t is periodic with a smaller period. By **(step 1)**, the periodic coloring contains all possible states 1, 2, …, $n^d k$. Thus, all cells at distance less than or equal to n corresponds to disjoined cells.

Step 5. Next step consists in proving a property of uniformity in time:



$$\forall p, \exists s_0, T_0, \forall t, \quad \mathbf{\Lambda}(t,p) = \{tT_0\} \times \left(\pi_2\left(\mathbf{\Lambda}(0,p)\right) + ts_0\right)$$

where π_2 projects sets of space-time cells to their space components. This property states that the successive images of a given cell are regularly aligned in space-time.

Let p be a position in space. Let $(p_1, \dots, p_k)$, $(p'_1, \dots, p'_k)$ and t be such that $\mathbf{\Lambda}(0, p) = ((0, p_1), \dots, (0, p_k))$ and $\mathbf{\Lambda}(1, p) = ((t, p'_1), \dots, (t, p'_k))$.

Let $\mathcal{A}$ be the CA with state set $\{\perp, 1, \dots, k, W_1, \dots, W_{t-1}\}$, neighborhood radius $2 \max \{\|p_1\|_\infty, \dots, \|p_k\|_\infty, \|p'_1\|_\infty, \dots, \|p'_k\|_\infty\}$ and such that one of its space-time diagrams Δ is filled with state $\perp$ for all times between 0 and t but:

$$\begin{aligned} \Delta(0, p_1) = 1, \dots, \Delta(0, p_k) = k, \\ \Delta(1, 0) = W_1, \dots, \Delta(t-1, 0) = W_{t-1}, \\ \Delta(t, p'_1) = 1, \dots, \Delta(t, p'_k) = k. \end{aligned}$$

All undefined transitions are mapped to $\perp$. Let Δ' be the transformed diagram of Δ. By construction, at time steps 0 and 1, the configurations of Δ' are filled with $(\perp, \dots, \perp)$ but at position p where it is equal to $(1, \dots, k)$. As Δ' is a space-time diagram of a CA and as its configurations at time 0 and 1 are equal, all the configurations of Δ' are equal. By construction, it implies that all the cells are uniformly shifted: $\exists s,\ p_1 = p'_1 + s \ \wedge \ \dots \ \wedge \ p_k = p'_k + s$. The construction straightforwardly extends to all time steps.

Step 6. Previous property stands up to an elementary shift:

$$\forall i, p, \exists s_i, T_i, \forall t, \quad \mathbf{\Lambda}(t, p + t\varsigma_i) = \{tT_i\} \times (\pi_2(\mathbf{\Lambda}(0, p)) + ts_i)$$

Replay the same arguments as for previous property but considering translated cells at time 1.

Step 7. We can now conclude that $\mathbf{\Lambda} = \mathbf{PCS}_{\pi_2(\mathbf{\Lambda}(0,0)),(s_1-s_0,\dots,s_d-s_0),T_0,s_0}$ as elementary translations form a base for translations in $\mathbb{Z}^d$. ■

4 Axiomatics of Bulking Quasi-Orders

The basic ingredients of a bulking simulation are now clear: a set of objects with an elementary comparison relation and an algebra of (geometrical) transforms to apply on objects. Simulation is then defined by comparison up to

transformation on both sides. Strong simulation is defined by comparison up to transformation on the simulator only.

In this section, the basic ingredients of bulking are formalized and properties of quasi-ordering and strong universality are established. Then, a model of bulking based on most general space-time transforms is discussed. Finally, a first model of bulking based on rectangular transforms and the subautomaton relation is introduced.

4.1 Theory of Bulking

The properties being sufficiently elementary, we choose to present bulking in a very formal way, as a first-order theory, rather than using a more classical algebraic definition. The two points of view are equivalent. Grouping being a model of bulking, it will be used to illustrate formal stuff.

Definition 10 *The* bulking *is the theory* $\mathbf{\Phi_b}$ *on the two-sorted signature*

$$\Big(\mathsf{Obj}, \mathsf{Trans};\ \mathbf{apply} : \mathsf{Obj} \times \mathsf{Trans} \to \mathsf{Obj},\ \mathbf{divide} \subseteq \mathsf{Obj} \times \mathsf{Obj},\ \mathbf{combine} : \mathsf{Trans} \times \mathsf{Trans} \to \mathsf{Trans}\Big)$$

defined by the following axioms, where latin letters (x*,* y*, ...) denotes elements of the sort* Obj*, greek letters (*α*,* β*, ...) and number* 1 *denotes elements of the sort* Trans*,* x^α *denotes* $\mathbf{apply}(x, \alpha)$*,* $x \mid y$ *denotes* $\mathbf{divide}(x, y)$ *and* $\alpha \cdot \beta$ *denotes* $\mathbf{combine}(\alpha, \beta)$*:*

$$(B_1) \quad \exists 1 \forall \alpha\, (\alpha \cdot 1 = \alpha \wedge 1 \cdot \alpha = \alpha) \ \wedge\ \forall \alpha \forall \beta \forall \gamma\, ((\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma))$$

$$(B_2) \quad \forall x\, (x^1 = x) \ \wedge\ \forall x \forall \alpha \forall \beta \left((x^\alpha)^\beta = x^{\alpha \cdot \beta}\right)$$

$$(B_3) \quad \forall x\, (x \mid x) \ \wedge\ \forall x \forall y \forall z\, ((x \mid y \wedge y \mid z) \to x \mid z)$$

$$(B_4) \quad \forall x \forall y \forall \alpha\, (x \mid y \to x^\alpha \mid y^\alpha)$$

$$(B_5) \quad \forall \alpha \forall x \exists y\, (x \mid y^\alpha)$$

$$(B_6) \quad \forall \beta \exists \gamma \forall \alpha \exists \delta\, (\alpha \cdot \gamma = \beta \cdot \delta)$$

Definition 11 *The* simulation relation $x \preccurlyeq y$ *is syntactically defined for all* $x, y \in \mathsf{Obj}$ *by the formula* $\exists \alpha \exists \beta \left(x^\alpha \mid y^\beta\right)$*.*

Objects, transforms and relations between them can be visualized graphically by considering elements of Obj as vertices and two kind of edges: a wiggly edge labelled by an element of Trans represents **apply**, a regular edge represents the relation **divide**. The simulation relation is depicted on Fig. 3.

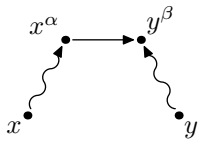


Fig. 3. Visual representation of the simulation relation

Grouping is a model of bulking where Obj is the set of global rules of 1D CA with neighborhood $\{-1, 0, 1\}$, Trans is the set of square transforms $\mathbb{N}$, **apply**(G, n) is the global rule $\square^n \circ G^n \circ \square^{-n}$, **divide** is the subautomaton relation $\sqsubseteq$ and **combine** is the classical arithmetical product on $\mathbb{N}$.

Axioms of $\mathbf{\Phi_b}$ formalize necessary algebraic properties:

(B_1) The structure $(\mathsf{Trans}, \cdot)$ is a monoid.
(B_2) The operator **apply** is an action of the monoid $(\mathsf{Trans}, \cdot)$ on the set Obj.
(B_3) The relation **divide** is a quasi-order on Obj.
(B_4) The operator **apply** is compatible with **divide**.
(B_5) The operator **apply** preserves the diversity of objects.
(B_6) The monoid $(\mathsf{Trans}, \cdot)$ admits a strong diamond property.

The (B_5) axiom ensures transforms do not pathologically weakens the elementary simulation relation.

The (B_6) axiom might seem less natural. In fact, in the case of grouping, the square transforms commute and (B_6) could be replaced by a commutation axiom $\forall\alpha\forall\beta\,(\alpha \cdot \beta = \beta \cdot \alpha)$. Due to the shift transform, even rectangular shifted transforms do not commute but satisfy the strong diamond property of (B_6). The simulation relation is a quasi-order given only (B_{1-5}) if and only if it satisfies $\forall x\forall\alpha\forall\beta\exists\gamma\exists\delta\left(x^{\alpha\cdot\gamma} \mid x^{\beta\cdot\delta}\right)$. By letting γ depend only on x and β one can derive Thm. 13 on strongly universal objects.

Theorem 12 *The simulation relation is a quasi-order, formally:*

$$\mathbf{\Phi_b} \vdash \forall x\,(x \preccurlyeq x) \;\wedge\; \forall x\forall y\forall z\,((x \preccurlyeq y \wedge y \preccurlyeq z) \to x \preccurlyeq z)$$

PROOF. The simulation relation is reflexive: $\mathbf{\Phi_b} \vdash \forall x\,(x \preccurlyeq x)$. By combining ($B_2$) and ($B_3$), it holds $\mathbf{\Phi_b} \vdash \forall x\,(x^1 \mid x^1)$.

The simulation relation is transitive: $\mathbf{\Phi_b} \vdash \forall x\forall y\forall z\,((x \preccurlyeq y \wedge y \preccurlyeq z) \to x \preccurlyeq z)$. Let x, y, z and α, β, γ, δ be such that $x^\alpha \mid y^\beta$ and $y^\gamma \mid z^\delta$. By (B_6) there exists η and ν such that $\beta \cdot \eta = \gamma \cdot \nu$. By ($B_4$) and ($B_2$), $x^{\alpha\cdot\eta} \mid y^{\beta\cdot\eta}$ and $y^{\gamma\cdot\nu} \mid z^{\delta\cdot\nu}$. Thus, by ($B_3$), $x^{\alpha\cdot\eta} \mid z^{\delta\cdot\nu}$, implying $x \preccurlyeq z$. ■

Theorem 13 *If a strongly universal object exists then all universal objects*

are strongly universal, formally:

$$\mathbf{\Phi_b} \vdash \exists u \forall x \exists \alpha \, (x \mid u^\alpha) \rightarrow \forall x \left(\forall y \, (y \preccurlyeq x) \rightarrow \forall y \exists \beta \left(y \mid x^\beta \right) \right)$$

PROOF. Let u be a strongly universal object: $\forall x \exists \alpha \, (x \mid u^\alpha)$. Let x be a universal object: $\forall y \, (y \preccurlyeq x)$. By universality of x, there exists α and β such that $u^\alpha \mid x^\beta$. By (B_6), there exists γ such that $\forall \eta \exists \delta \, (\eta \cdot \gamma = \alpha \cdot \delta)$. Let y be an object. By (B_5), there exists z such that $y \mid z^\gamma$. By strong universality of u, there exists η such that $z \mid u^\eta$. By (B_4) and (B_2), $z^\gamma \mid u^{\eta \cdot \gamma}$. Let δ be such that $\eta \cdot \gamma = \alpha \cdot \delta$. By (B_4) and (B_2), $u^{\alpha \cdot \delta} \mid x^{\beta \cdot \delta}$. Thus, by (B_3), $y \mid x^{\beta \cdot \delta}$. ■

4.2 CA Bulking: A First Attempt

Thanks to previous discussion, one might try to build a model of bulking using d-CA as sort Obj, **PCS** transforms as sort Trans (following Thm. 9) and the subautomaton relation $\sqsubseteq$ as elementary simulation relation (following Thm. 4). However, axiom (B_6) is not satisfied by **PCS** transforms. More precisely, the problem roots in the structure of the monoid of composition of tilings of space which does not admit the diamond property.

Example 14 *Let $\mathcal{P}_1 = \{0, 1\}$, $\mathcal{P}_2 = \{0, 3\}$ and $v = 2$. Let Θ be the set of all d-tilings of space. The tilings of space $\langle \mathcal{P}_1, v \rangle$ and $\langle \mathcal{P}_2, v \rangle$, depicted on Fig. 4, are incompatible: $(\Theta \circ \langle \mathcal{P}_1, v \rangle) \cap (\Theta \circ \langle \mathcal{P}_2, v \rangle) = \emptyset$.* ◇

Fig. 4. Two incompatible tilings of space

Notice that the problem does not come from the composition of bases themselves (product of non-zero determinant matrices with integer coefficients admits a strong diamond property) but really from the geometrical shape of tilings. Intuitively, in Example 14, compositions of $\mathcal{P}_1$ always contains two consecutive elements on the extreme left whereas compositions of $\mathcal{P}_2$ contains a single element followed by a gap. Starting from dimension 2, even connected tilings leads to problem, shapes replacing gaps.

Open Problem 1 *Characterize the submonoids of composition of tilings of space that admits the diamond property.*

As we want to extend grouping, we need to select a submonoid with the strong diamond property that contains square transformations. The set of rectangular tilings constitutes an adequate commutative submonoid. One might enrich a bit this set by allowing permutation and negation of elements of the basis.

Rectangular Packing. Let $m \in (\mathbb{Z}^+)^d$ and $\tau \in \mathbb{Z}^d$ such that τ is a signed permutation of $(1, \ldots, d)$. The *rectangular packing* $\tilde{\mathbf{P}}_{m,\tau}$ is the packing $\mathbf{P}_{\boxplus_m, V_\tau \odot \square_m}$ where V_τ is the basis where $V_\tau(k)$ has all its elements equals to 0 but in position $|\tau_k|$ where it is equal to 1 if $\tau_k > 0$ and to -1 if $\tau_k < 0$.

Composition. Let (m, τ) be valid rectangular packing parameters, s be a translation vector and T be a rate, the nice geometrical transform $\tilde{\mathbf{PCS}}_{m,\tau,T,s}$ is the composition $\tilde{\mathbf{P}}_{m,\tau} \circ \mathbf{S}_s \circ \mathbf{C}_T$. The set of $\tilde{\mathbf{PCS}}$ transforms generated by pure $\tilde{\mathbf{P}}$, $\mathbf{C}$ and $\mathbf{S}$ transforms is closed by composition:

$$\tilde{\mathbf{PCS}}_{(m'_1,\ldots,m'_d),\tau',T',s'} \circ \tilde{\mathbf{PCS}}_{(m_1,\ldots,m_d),\tau,T,s} = \tilde{\mathbf{PCS}}_{(m''_1,\ldots,m''_d),\tau'',T'',s''}$$

with parameters

$$\begin{aligned}
m'' &= \varrho(\tau, m, m') \quad \text{where } \varrho(\tau, m, m')_i = m_i m'_{|\tau(i)|} \\
\tau'' &= \tau' \otimes \tau \quad \text{where } (\tau' \otimes \tau)_i = \mathrm{sg}(\tau'_i) \times \tau_{|\tau'_i|} \\
T'' &= TT' \\
s'' &= (s' \odot V_\tau \odot \square_m) + T's
\end{aligned}$$

Let τ^{-1} denote the inverse of τ with respect to $\otimes$, that is such that $\tau^{-1} \otimes \tau = \mathrm{id}$ where $\mathrm{id} = (1, \ldots, d)$.

To simplify notations, in the rest of the paper, $\langle m, \tau, T, s \rangle$ denotes a valid $\tilde{\mathbf{PCS}}_{m,\tau,T,s}$ transform and product on this notation denotes composition. Notice that $\langle 1, \tau^{-1}, 1, 0 \rangle \langle m, \tau, T, s \rangle = \langle m, \mathrm{id}, T, s \rangle$.

Composition of $\tilde{\mathbf{PCS}}$ is quite symmetrical but for the shifting component.

Lemma 15 $\tilde{\mathbf{PCS}}$ *transforms have the strong diamond property, that is* (B_6).

PROOF. Given a $\tilde{\mathbf{PCS}}$ transform $\beta = \langle m, \tau, T, s \rangle$, let

$$\gamma = \langle \mathrm{lcm}(m)(1, \ldots, 1), \mathrm{id}, \mathrm{lcm}(m)T, 0 \rangle \quad .$$

For all $\alpha = \langle m', \tau', T', s' \rangle$, let $\delta = \langle \widetilde{m}, \tau' \otimes \tau^{-1}, \mathrm{lcm}(m)T', \widetilde{s} \rangle$. By definition,

$$\begin{aligned}
\alpha \cdot \gamma &= \langle \mathrm{lcm}(m)m', \tau', \mathrm{lcm}(m)TT', \mathrm{lcm}(m)Ts' \rangle \\
\beta \cdot \delta &= \langle \varrho(\tau, m, \widetilde{m}), \tau', \mathrm{lcm}(m)TT', \mathrm{lcm}(m)T's + (\widetilde{s} \odot V_\tau \odot \square_m) \rangle
\end{aligned}$$

As each component of $\mathrm{lcm}(m)m'$ is a mutiple of each component of m, one can choose $\widetilde{m}$ such that $\varrho(\tau, m, \widetilde{m}) = \mathrm{lcm}(m)m'$. As each component of $\mathrm{lcm}(m)Ts'$ is a multiple of each component of m, one can choose $\widetilde{s}$ such that $\widetilde{s} \odot V_\tau \odot \square_m = \mathrm{lcm}(m)(Ts' - T's)$. ■

4.3 CA Bulking: A Model

Definition 16 *Let $\mathcal{A}$ and $\mathcal{B}$ be two d-CA. $\mathcal{B}$ simulates $\mathcal{A}$ injectively, denoted $\mathcal{A} \preccurlyeq_{\mathrm{i}} \mathcal{B}$, if there exists two $\tilde{\mathsf{PCS}}$ transforms $\alpha = \langle m, \tau, T, s\rangle$ and $\beta = \langle m', \tau', T', s'\rangle$ such that the transform of $\mathcal{A}$ by α is a subautomaton of the transform of $\mathcal{B}$ by β. Formally,*

$$\langle \boxplus_m, V_\tau \odot \square_m\rangle \circ \sigma_s \circ G_{\mathcal{A}}^T \circ \langle \boxplus_m, V_\tau \odot \square_m\rangle^{-1} \sqsubseteq \langle \boxplus_{m'}, V_{\tau'} \odot \square_{m'}\rangle \circ \sigma_{s'} \circ G_{\mathcal{B}}^{T'} \circ \langle \boxplus_{m'}, V_{\tau'} \odot \square_{m'}\rangle^{-1} \quad .$$

Theorem 17 *The set of d-CA equipped with $\tilde{\mathsf{PCS}}$ transforms and the subautomaton relation $\sqsubseteq$ is a model of bulking.*

PROOF. Each axiom has to be checked:

Axiom (B_1). $(\tilde{\mathsf{PCS}}, \circ)$ is a monoid with unit $\langle (1, \ldots, 1), \mathrm{id}, 1, 0\rangle$.

Axiom (B_2). By definition, applying $\tilde{\mathsf{PCS}}$ transforms is an action of CA.

Axiom (B_3). The subautomaton relation is a straightforward quasi-order: $\mathcal{A} \subseteq_{\mathrm{id}} \mathcal{A}$ and if $\mathcal{A} \subseteq_\varphi \mathcal{B}$ and $\mathcal{B} \subseteq_\psi \mathcal{C}$ then $\mathcal{A} \subseteq_{\psi\circ\varphi} \mathcal{C}$.

Axiom (B_4). The subautomaton relation is compatible with $\tilde{\mathsf{PCS}}$ transform by product extension of injective φ functions: if $\mathcal{A} \sqsubseteq_\varphi \mathcal{B}$ then $\mathcal{A}^{\langle m,\ldots\rangle} \sqsubseteq_\psi \mathcal{B}^{\langle m,\ldots\rangle}$ where $\psi((s_1, \ldots, s_{\prod m_i})) = (\varphi(s_1), \ldots, \varphi(s_{\prod m_i}))$.

Axiom (B_5). The application of $\tilde{\mathsf{PCS}}$ transforms preserves the diversity of objects: let $\langle m, \tau, T, s\rangle$ be a $\tilde{\mathsf{PCS}}$ transform and let $\mathcal{A} = (S, N, f)$ be a CA. Let $\mathcal{B} = (S, N', f')$ where $N' = V_\tau \odot \square_m \odot N$ and for all $a \in S^N$, for all $f'(V_\tau \odot \square_m \odot a)(z) = f(a)$. By construction, $\mathcal{A} \sqsubseteq_\varphi \mathcal{B}^{\langle m,\tau,T,s\rangle}$ where $\varphi(s) = (s, \ldots, s)$.

Axiom (B_6). This is Lemma 15. ■

Corollary 18 $(\mathrm{CA}, \preccurlyeq_{\mathrm{i}})$ *is a quasi-ordered set.*

PROOF. This is Theorem 12. ■

5 CA Bulking and Universality

In this section, we investigate elementary properties of the CA bulking introduced in previous section, compare it to grouping and obtain the first results on bulking, in particular with respect to intrinsic universality. This bulking, as well as others, are studied more in depth in *Bulking II: Classifications of Cellular Automata* [4].

Definition 19 *A CA $\mathcal{U}$ is* intrinsically universal *if it strongly simulates injectively every CA: for all CA $\mathcal{A}$, there exists a $\tilde{\mathsf{PCS}}$ transform α such that $\mathcal{A} \sqsubseteq \mathcal{U}^\alpha$.*

Theorem 20 *Every maximal element of $\preccurlyeq_{\mathrm{i}}$ is intrinsically universal.*

PROOF. This is a consequence of Theorem 13. ■

Theorem 21 *There exists no real-time intrinsically universal CA.*

PROOF. This is a consequence of Theorem 3. ■

References

[1] J. Albert, K. Čulik, II, A simple universal cellular automaton and its one-way and totalistic version, Complex Systems 1 (1) (1987) 1–16.

[2] E. R. Banks, Universality in cellular automata, in: Symposium on Switching and Automata Theory (Santa Monica, California, 1970), IEEE, 1970, pp. 194–215.

[3] A. W. Burks (ed.), Essays on cellular automata, University of Illinois Press, Urbana, Ill., 1970.

[4] M. Delorme, J. Mazoyer, N. Ollinger, N. Theyssier, Bulking II: Classifications of cellular automata, *unpublished* (2008).

[5] P. C. Fischer, Generation of primes by a one-dimensional real-time iterative array, Journal of the ACM 12 (3) (1965) 388–394.

[6] G. A. Hedlund, Endormorphisms and automorphisms of the shift dynamical system, Mathematical Systems Theory 3 (1969) 320–375.

[7] G. A. Hedlund, Endormorphisms and automorphisms of the shift dynamical system, Mathematical Systems Theory 3 (1969) 320–375.

[8] D. Lind, B. Marcus, An introduction to symbolic dynamics and coding, Cambridge University Press, Cambridge, 1995.

[9] B. Martin, A universal cellular automaton in quasi-linear time and its s-m-n form, Theor. Comput. Sci. 123 (2) (1994) 199–237.

[10] B. Martin, A geometrical hierarchy on graphs via cellular automata, Fundam. Inform. 52 (1-3) (2002) 157–181.

[11] J. Mazoyer, I. Rapaport, Additive cellular automata over Z_p and the bottom of $(\mathrm{ca}, \leq)$, in: Mathematical foundations of computer science (Brno, 1998), Springer, Berlin, 1998, pp. 834–843.

[12] J. Mazoyer, I. Rapaport, Inducing an order on cellular automata by a grouping operation, Discrete Applied Mathematics 91 (1-3) (1999) 177–196.

[13] Z. Róka, Cellular automata on cayley graphs, Ph.D. thesis, École Normale SupÃľrieure de Lyon (1994).

[14] J. von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, Ill., 1966.

A.2 Bulking II : Classifications of Cellular Automata

Version courante de [U2], seconde partie d'une série de deux articles écrite en collaboration avec M. Delorme, J. Mazoyer and G. Theyssier. Cet article et le précédent font la synthèse des travaux sur le groupage réalisés à Lyon à partir de la thèse fondatrice d'I. Rapaport [80] dont les principaux résultats ont été publiés par J. Mazoyer et I. Rapaport dans [63] et [65]. Plus précisément, ils synthétisent les résultats présentés en français dans les thèse de N. Ollinger [T1] et G. Theyssier [88].

Bulking II: Classifications of Cellular Automata

M. Delorme[a], J. Mazoyer[a], N. Ollinger[b], G. Theyssier[c,*]

[a] *LIP, ENS Lyon, CNRS, 46 allée d'Italie, 69 007 Lyon, France*
[b] *LIF, Aix-Marseille Université, CNRS, 39 rue Joliot-Curie, 13 013 Marseille, France*
[c] *LAMA, Université de Savoie, CNRS, 73 376 Le Bourget-du-Lac Cedex, France*

Abstract

This paper is the second part of a serie of two papers dealing with bulking: a quasi-order on cellular automata comparing space-time diagrams up to some rescaling. Bulking is a generalization of grouping taking into account universality phenomena, giving rise to a maximal equivalence class. In the present paper, we introduce 3 notions of simulation between cellular automata and study the quasi-order structures induced by these simulation relations on the whole set of cellular automata. Various aspects of these quasi-orders are considered (induced equivalence relations, maximum elements, induced orders, etc) providing several formal tools to classify cellular automata.

Key words: cellular automata, bulking, grouping, classification

1. Introduction

In the first paper [7], we have seen how the relation "to be a subautomaton" induces an order on the classes of automata (defined by general bulking). In the present paper we aim to study new relations also inducing an order. The two "natural" relations are "to be a subautomaton", $\preceq_i$, and "to be a quotient", $\preceq_s$. Both induce orders and any mix of them in included in "to be a projection of a subautomaton", $\preceq_m$. So, we get three notions of simulation and three orders on classes. A cellular automaton $\mathcal{A}$ is $\preceq_i$-simulated by $\mathcal{B}$ if every orbit of $\mathcal{A}$ is an orbit of $\mathcal{B}$; so every global property of $\mathcal{A}$ is a global property of $\mathcal{B}$. A cellular automaton $\mathcal{A}$ is $\preceq_s$-simulated by $\mathcal{B}$ if every orbit of $\mathcal{B}$ may be viewed as an orbit of $\mathcal{A}$ when some states of $\mathcal{B}$ are interpreted as the same state; thus, every global property of $\mathcal{A}$ can be viewed and a refinement of a global property of $\mathcal{B}$ and every global property of $\mathcal{B}$ is a refinement of a global property of $\mathcal{A}$. Finally, a cellular automaton $\mathcal{A}$ is $\preceq_m$-simulated by $\mathcal{B}$ if every orbit of $\mathcal{A}$ is a projection of some orbit of $\mathcal{B}$; thus, every global property of $\mathcal{A}$ can be viewed as a refinement of a global property of $\mathcal{B}$.

As we handle global properties of cellular automata, these orders may be considered as classification of cellular automata. Historically, the empirical classification (human observation of orbits) and descriptive (propagating structures, computation ability, change of information, ...) in four classes proposed by S.

*Corresponding author (Guillaume.Theyssier@univ-savoie.fr)

Wolfram [36] have popularized cellular automata and have opened the problematic to classify cellular automata. First, to classify has no sense without additional assumptions (some criteria of classification). If in Wolframs papers these criteria were implicit, several classifications with explicit criteria have been proposed since. Usually, the criteria are those of dynamical systems. After works of Gilman [10], Cattaneo *et al* [3], a classification in four classes was proposed by Kůrka [18] based on the influence of local modification of a cell on the global dynamics. We observe that the topology on the set of initial configurations plays an important role: Cantor topology (in general) or Besicowich pseudo-topology (Formenti [9]). The point of view of Čulik seems to capture classes I and II of Wolfram and the dynamical classes (classes III and IV) of Kůrka captures class III of Wolfram. But Wolfram's class IV is not handled. This last class is the more interesting because it captures self-organisation and emergence of more of less regular structures ("particles", "parabolas", ...). In classifications induced by our three orders, classes I and II of Wolfram become classes immediately above the minimal class and the classification tends to capture Wolfram's class IV. For example, "to have a particle" is a class for $\preceq_i$-classification. Classes of reversible (or symmetrical, ...) cellular automata form an ideal in our orders.

We study the structure of these orders. At finite level (classes with a finite non zero number of classes above them), there is infinitely many classes. Orders are not upper or lower semi-lattices. For example, infinite increasing sequences may be due to the following fact: some integer may be coded by states or (at the limit) by extracting this integer from the data of the initial configuration. Orders $\preceq_i$ and $\preceq_m$ have a maximal class (intrinsically universal in [7]) but not $\preceq_s$. Classes having Turing-universality are obtained by simulating (in a way closed to Smith III [31]) an universal Turing machine. Such a class is not necessarily at the top (intrinsically universal). Moreover there exists a cellular automaton which, in some sense, handles potential infinite. Precisely, for $\preceq_m$-simulation, there exists an infinite increasing sequence of cellular automata (simulating n copies of a Turing machine but not $n+1$ copies) and the limit cellular automata $\preceq_m$-simulates any finite number of copies but not an infinite number. Every class contains a cellular automaton with radius 1 but, more surprising, there exist classes which do not contain cellular automata with two states of any neighborhood. This fact implies that some global properties needs more than two states to appear whatever is the neighborhood.

Finally, if in other formal classifications (Čulik, Kůrka) to belong to a class may be undecidable and relation between classes are decidable (classes are in finite number), in our orders, we show that both may be undecidable.

Section 1.1 introduces 3 different comparison relations which are 3 different instances of the bulking theory developed in the companion paper [7]. Section 2 sets the definitions of these 3 notions of simulations and establish some of their basic properties. Section 3 studies the 'bottom' of each of the 3 quasi-orders induced on CA, *i.e.* CA or classes of CA of least complexity. Section 4 focuses on the order structure with respect to various classical properties of CA, and from a computability point of view. Then section 5 explores the set of CA at the 'top' of these quasi-orders: universal CA. Once again, the point of view is both structural and computational. Finally, section 6 is devoted to the construction of noticeable induced orders (like infinite chains), and the study of how simple families of CA spread over those quasi-orders.

1.1. Definitions

In this paper, we adopt the setting of one-dimensional cellular automata with a canonical neighborhood (connected and centered). A cellular automaton (CA) is a triple $\mathcal{A} = (S, r, f)$ where:

- S is a (finite) *state set*,
- r is the neighborhood *radius*,
- $f : S^{2r+1} \to S$ is the *local transition function*.

A coloring of the lattice $\mathbb{Z}$ with states from S (*i.e.* an element of $S^{\mathbb{Z}}$) is called a *configuration*. To $\mathcal{A}$ we associate a global function G acting on configurations by synchronous and uniform application of the local transition function. Formally, $G : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ is defined by:

$$G(x)_z = f(x_{z-r}, \ldots, x_{z+r})$$

for all $z \in \mathbb{Z}$. Several CA can share the same global function although there are syntactically different (different radii and local functions). However we are mainly interested in global functions and will sometimes define CA through their global function without specifying a particular syntactical representation. In addition, the Curtis-Heldund-Lyndon theorem [13] allows us to freely compose global CA functions to construct new CA without manipulating explicitly the underlying syntactical representation.

When dealing with several CA simultaneously, we use index notation to denote their respective state sets, radii and local functions. For instance, to $\mathcal{A}$ we associate $S_{\mathcal{A}}$, $r_{\mathcal{A}}$ and $f_{\mathcal{A}}$.

This paper will make an intensive use of $\tilde{\mathbf{P}}\mathbf{CS}$ transforms defined in section 4.2 of [7], but restricted to dimension 1. With this restriction, a $\tilde{\mathbf{P}}\mathbf{CS}$ transform α has the form $\alpha = \langle m, \tau, T, s\rangle$ where m and T are positive integers, s is a (possibly negative) integer and τ is either 1 or -1.

For any CA $\mathcal{A}$, we denote by $\mathcal{A}^{\langle\alpha\rangle}$ or more explicitly $\mathcal{A}^{\langle m,\tau,T,s\rangle}$ the application of α to $\mathcal{A}$, which is, according to notations of [7], a CA of state set $S_{\mathcal{A}}^m$ and global rule:

$$\langle \boxplus_m, V_\tau \odot \square_m\rangle \circ \sigma_s \circ G_{\mathcal{A}}^T \circ \langle \boxplus_m, V_\tau \odot \square_m\rangle^{-1}.$$

To simplify notation we will use a shortcut for purely temporal transforms: for any CA $\mathcal{A}$ we denote by $\mathcal{A}^t$ the CA $\mathcal{A}^{\langle 1,1,t,0\rangle}$. Finally, as another special case, we denote by $\mathcal{A}^{[n]}$ the grouped instance of $\mathcal{A}$ of parameter n: it corresponds to the transform $\langle n, 1, n, 0\rangle$ (see [7] for a detailed exposition of grouping).

2. Canonical orders

In this section we introduce the 3 bulking quasi-orders that are studied all along the paper. They are obtained by applying the bulking axiomatics developed in the companion paper [7] to 3 'canonical' relations between local rules of CA.

Those 3 'canonical' relations are in turn based on 2 classical notions of morphism between local transition rules of CA: sub-automaton and quotient-automaton. As it is shown below, the 3 relations we consider are exactly the reflexive and transitive relations that can be defined by composition of one or more such morphisms.

2.1. From 3 Local Relations to 3 Bulking Quasi-Orders

A *sub-automaton* is a restriction of a CA to a stable sub-alphabet. A *quotient* is a projection of a CA onto a smaller alphabet and compatible with the local transition rule[1]. Both define a kind of morphism between cellular automata:

- $\mathcal{A}$ is a *sub-automaton* of $\mathcal{B}$, denoted $\mathcal{A} \sqsubseteq \mathcal{B}$, if there is an injective map $\iota : S_{\mathcal{A}} \to S_{\mathcal{B}}$ such that $\overline{\iota} \circ G_{\mathcal{A}} = G_{\mathcal{B}} \circ \overline{\iota}$, where $\overline{\iota} : S_{\mathcal{A}}^{\mathbb{Z}} \to S_{\mathcal{B}}^{\mathbb{Z}}$ denotes the uniform extension of ι. We often write $\mathcal{A} \sqsubseteq_{\iota} \mathcal{B}$ to make the map ι explicit.
- $\mathcal{A}$ is a *quotient* of $\mathcal{B}$, denoted $\mathcal{A} \trianglelefteq \mathcal{B}$, if there is a surjective (onto) map s from $S_{\mathcal{B}}$ to $S_{\mathcal{A}}$ such that $\overline{s} \circ G_{\mathcal{B}} = G_{\mathcal{A}} \circ \overline{s}$, where $\overline{s} : S_{\mathcal{B}}^{\mathbb{Z}} \to S_{\mathcal{A}}^{\mathbb{Z}}$ denotes the uniform extension of s. We also write $\mathcal{A} \trianglelefteq_{s} \mathcal{B}$ to make the map s explicit.

Relations $\sqsubseteq$ and $\trianglelefteq$ are quasi-orders (reflexive and transitive) and it is straightforward to check that their induced equivalence relation is the relation of isomorphism between cellular automata (equality up to state renaming) denoted by $\equiv$.

It is also straightforward to check that $\sqsubseteq$ and $\trianglelefteq$ are incomparable (none of them is implied by the other). It is thus interesting to consider compositions of them. The composition of two relations R_1 and R_2 is the relation $R_1 \cdot R_2$ defined by

$$R_1 \cdot R_2 = \{(x, y) : \exists z, (x, z) \in R_1 \text{ and } (z, y) \in R_2\}.$$

We denote by $\mathcal{R}$ the set of relations obtained by (finite) composition of $\trianglelefteq$ and $\sqsubseteq$. Any relation of $\mathcal{R}$ is a priori interesting, but the following theorem justifies that we restrict to $\trianglelefteq$, $\sqsubseteq$ and the composition $\trianglelefteq \cdot \sqsubseteq$ only. In the sequel $\trianglelefteq \cdot \sqsubseteq$ is denoted by $\trianglelefteq\!\!\sqsubseteq$ and, as for $\sqsubseteq$ and $\trianglelefteq$, we use the infix notation ($\mathcal{A} \trianglelefteq\!\!\sqsubseteq \mathcal{B}$).

Theorem 2.1.

1. *any relation $R \in \mathcal{R}$ is included in $\trianglelefteq\!\!\sqsubseteq$ (i.e., $(\mathcal{A}, \mathcal{B}) \in R$ implies $\mathcal{A} \trianglelefteq\!\!\sqsubseteq \mathcal{B}$) ;*
2. *the transitive relations of $\mathcal{R}$ are exactly: $\trianglelefteq$, $\sqsubseteq$ and $\trianglelefteq\!\!\sqsubseteq$.*

Proof. We first prove that if $\mathcal{A} \sqsubseteq \cdot \trianglelefteq \mathcal{B}$ then $\mathcal{A} \trianglelefteq\!\!\sqsubseteq \mathcal{B}$, which is sufficient to prove (1) by transitivity of $\sqsubseteq$ and of $\trianglelefteq$. So consider $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ such that $\mathcal{A} \sqsubseteq_{\iota} \mathcal{C}$ and $\mathcal{C} \trianglelefteq_{s} \mathcal{B}$. Then consider $Q = s^{-1} \circ \iota(S_{\mathcal{A}})$. We have $G_{\mathcal{B}}(Q^{\mathbb{Z}}) \subseteq Q^{\mathbb{Z}}$ because

$$\overline{s} \circ G_{\mathcal{B}}(Q^{\mathbb{Z}}) = G_{\mathcal{C}} \circ \overline{s}(Q^{\mathbb{Z}}) = G_{\mathcal{C}} \circ \overline{\iota}(S_{\mathcal{A}}^{\mathbb{Z}}) = \overline{\iota} \circ G_{\mathcal{A}}(S_{\mathcal{A}}^{\mathbb{Z}}) \subseteq \overline{\iota}(S_{\mathcal{A}}^{\mathbb{Z}}).$$

The CA $\mathcal{X} = (Q, r_{\mathcal{B}}, f_{\mathcal{B}})$ is thus well-defined and by definition we have $\mathcal{X} \sqsubseteq \mathcal{B}$. Moreover, we have $\mathcal{A} \trianglelefteq_{\iota^{-1} \circ s} \mathcal{X}$ because $\iota^{-1} \circ s : Q \to A$ is well-defined and surjective and because

$$\overline{\iota^{-1} \circ s} \circ G_{\mathcal{X}} = G_{\mathcal{A}} \circ \overline{\iota^{-1} \circ s}$$

since $\overline{s} \circ G_{\mathcal{B}} = G_{\mathcal{C}} \circ \overline{s}$ and $\overline{\iota^{-1}} \circ G_{\mathcal{C}} = G_{\mathcal{A}} \circ \overline{\iota^{-1}}$ over $\big(\iota(A)\big)^{\mathbb{Z}} = \overline{s}\big(Q^{\mathbb{Z}}\big)$. Hence $\mathcal{A} \trianglelefteq\!\!\sqsubseteq \mathcal{B}$ and (1) is proven.

[1] A quotient is a particular kind of *factor*, a classical notion in dynamical systems theory and symbolic dynamics [19]

Given (1) we have $\mathcal{R} = \{\trianglelefteq, \sqsubseteq, \sqsubseteq \cdot \trianglelefteq, \trianglelefteq\!\!\sqsubseteq\}$. To prove (2), it is thus sufficient to prove that $\sqsubseteq \cdot \trianglelefteq$ is not transitive. To do this, consider $S_{\mathcal{A}} = \{0, \ldots, p-1\}$ with p prime, $p \geq 5$, and let $\alpha, a_0, a_1, b_0, b_1$ be 5 distinct elements of $S_{\mathcal{A}}$. Then consider $\mathcal{A}$, the CA of states set $S_{\mathcal{A}}$, radius 1 and local rule $f_{\mathcal{A}}$ defined by:

$$f_{\mathcal{A}}(*, x, y) = \begin{cases} a_{1-i} & \text{if } x \neq \alpha \text{ and } y = a_i, \\ b_{1-i} & \text{if } x \neq \alpha \text{ and } y = b_i, \\ y + 1 \bmod p & \text{else.} \end{cases}$$

$f_{\mathcal{A}}$ depends only on two variables. Suppose now that there is some AC $\mathcal{B}$ with at least 2 states such that $\mathcal{B} \trianglelefteq_\pi \mathcal{A}$. We will show that π must be one-to-one. Suppose indeed by contradiction that there are distinct elements e and f in $S_{\mathcal{A}}$ such that $\pi(e) = \pi(f)$. Then $\pi(e + 1 \bmod p) = \pi(f + 1 \bmod p)$ and more generally $\pi(e + i \bmod p) = \pi(f + i \bmod p)$ for all $i \in \mathbb{N}$ because $f_{\mathcal{A}}(\alpha, e) = e + 1 \bmod p$ and $f_{\mathcal{A}}(\alpha, f) = f + 1 \bmod p$. So, supposing without loss of generality $f > e$, let $k = f - e$. We deduce from above that $\pi(f) = \pi(f + jk \bmod p)$ for all $k \in \mathbb{N}$ and, by elementary group theory, that π is constant equal to $\pi(f)$ (because p is prime and $k \neq 0$). This is in contradiction with the fact that π is surjective onto $S_{\mathcal{B}}$ which has at least 2 elements. So π is one-to-one and $\mathcal{B}$ is isomorphic to $\mathcal{A}$. Now consider $\mathcal{C}$, the identity CA over state set $S_{\mathcal{C}} = \{0, 1\}$. Since $\mathcal{C}$ possesses 2 quiescent states and $\mathcal{A}$ has no quiescent state (straightforward from the definition of $f_{\mathcal{A}}$ above), we have $\mathcal{C} \not\trianglelefteq \mathcal{A}$. With the discussion above, we can conclude that $\mathcal{C} \sqsubseteq \cdot \not\trianglelefteq \mathcal{A}$.

However, we have $\mathcal{B} \trianglelefteq\!\!\sqsubseteq \mathcal{A}$ because the states $\{a_0, a_1, b_0, b_1\}$ induce a sub-automaton $\mathcal{C}$ of $\mathcal{A}$ which verifies $\mathcal{B} \trianglelefteq_s \mathcal{C}$ where $s : \{a_0, a_1, b_0, b_1\} \to \{0, 1\}$ is defined by $s(a_i) = 0$ et $s(b_i) = 1$. (2) follows since the relation $\trianglelefteq\!\!\sqsubseteq$ is included in the composition of the relation $\sqsubseteq \cdot \trianglelefteq$ with itself. ■

Like $\sqsubseteq$ (already considered in [7]), $\trianglelefteq$ and $\trianglelefteq\!\!\sqsubseteq$ are quasi-orders on CA and therefore constitute natural candidates for the **divide** relation of bulking axiomatics (definition 8 of [7]).

Inspired by definition 14 of [7], we now define 3 bulking quasi-orders using $\tilde{\mathbf{P}}\mathbf{C}\mathbf{S}$ transforms.

Definition 2.1. *$\mathcal{B}$ simulates $\mathcal{A}$* injectively*, denoted $\mathcal{A} \preccurlyeq_i \mathcal{B}$, if there exists two $\tilde{\mathbf{P}}\mathbf{C}\mathbf{S}$ transforms α and β such that $\mathcal{A}^{\langle\alpha\rangle} \sqsubseteq \mathcal{B}^{\langle\beta\rangle}$.*

We will occasionally use the notion of simulation by grouping introduced in [23] and discussed in [7]: we denote by $\mathcal{A} \leqslant_\square \mathcal{B}$ the fact that there is n and m such that $\mathcal{A}^{[n]} \sqsubseteq \mathcal{B}^{[m]}$. This is a special case of the injective simulation above.

Definition 2.2. *$\mathcal{B}$ simulates $\mathcal{A}$* surjectively*, denoted $\mathcal{A} \preccurlyeq_s \mathcal{B}$, if there exists two $\tilde{\mathbf{P}}\mathbf{C}\mathbf{S}$ transforms α and β such that $\mathcal{A}^{\langle\alpha\rangle} \trianglelefteq \mathcal{B}^{\langle\beta\rangle}$.*

Definition 2.3. *$\mathcal{B}$ simulates $\mathcal{A}$ in a mixed way, denoted $\mathcal{A} \preccurlyeq_m \mathcal{B}$, if there exists two $\tilde{\mathbf{P}}\mathbf{C}\mathbf{S}$ transforms α and β such that $\mathcal{A}^{\langle\alpha\rangle} \trianglelefteq\!\!\sqsubseteq \mathcal{B}^{\langle\beta\rangle}$*

For each notion of simulation above, we say that the simulation is *strong* if the transformation α applied to the simulated CA is trivial: $\alpha = \langle 1, 1, 1, 0\rangle$ so that $\mathcal{A}^{\langle\alpha\rangle} = \mathcal{A}$.

Theorem 2.2. *$\big(CA, \preccurlyeq_i\big)$, $\big(CA, \preccurlyeq_s\big)$ and $\big(CA, \preccurlyeq_m\big)$ are quasi-orders.*

PROOF. We show that $\preccurlyeq_i$ and $\preccurlyeq_m$ correspond exactly to models of bulking developped in [7]: the proof of theorem 15 of [7] contains the case of injective simulation. The case of $\preccurlyeq_m$ follows immediately (axiom (B_4) is straightforward and axiom (B_5) is true because $\trianglelefteq$ contains $\sqsubseteq$). For $\preccurlyeq_s$, the proof of each axiom is similar except for axiom (B_5).

With or without axiom (B_5), theorem 10 of [7] can be applied in each case and show the theorem. ■

In the sequel, if $\preccurlyeq$ denotes a simulation quasi-order we denote by $\sim$ the induced equivalence relation and by $[\mathcal{A}]$ the equivalence class of $\mathcal{A}$ with respect to $\sim$. For instance, to $\preccurlyeq_i$ we associate the notations $\sim_i$ and $[\mathcal{A}]_i$. We use similar notations for $\preccurlyeq_s$ and $\preccurlyeq_m$.

Before entering into details concerning various aspects of the 3 simulation relations defined above, we can already make a clear (yet informal) distinction between $\preccurlyeq_i$ and $\preccurlyeq_m$ on one hand, and $\preccurlyeq_s$ on the other hand. For the two former, the simulation take place on a subset of configurations and nothing can be said *a priori* about the behavior of the simulator outside this subset of configurations. For $\preccurlyeq_s$, however, the simulation occur on any configuration and the simulator's behavior on any configuration is in some way affected by the simulation. Section 4.2 give several illustrations of this difference.

2.2. First Properties

We now establish a set of basic general facts about $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$ while next sections of the paper focus on particular aspects.

Theorem 2.3. *Let $\mathcal{A}$ be any CA and $\preccurlyeq$ be any relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Then we have:*

1. *there is some $\mathcal{B} \in [\mathcal{A}]$ having a quiescent state,*
2. *there is some $\mathcal{B} \in [\mathcal{A}]$ with radius 1,*
3. *$\perp \preccurlyeq \mathcal{A}$ where $\perp$ is the CA with a single state,*
4. *$\mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{B}$ and $\mathcal{A} \preccurlyeq \mathcal{B} \times \mathcal{A}$ for any $\mathcal{B}$.*

PROOF.

1. there exists some uniform configuration x and some $t \geq 1$ such that $G^t_{\mathcal{A}}(x) = x$. So $\mathcal{A}^t$ has a quiescent state and it clearly belongs to $[\mathcal{A}]$.
2. $\mathcal{A}^{\langle r_{\mathcal{A}},1,1,0\rangle}$ admits a syntactical representation with radius 1 and clearly belongs to $[\mathcal{A}]$.
3. First, one always has $\perp \trianglelefteq_\pi \mathcal{A}$ where π is the trivial surjection sending each state of $\mathcal{A}$ to the single state of $\perp$. So assertion 3 is proven for $\preccurlyeq_s$ and $\preccurlyeq_m$. Second, one has $\perp \sqsubseteq_i \mathcal{B}$ if $\mathcal{B}$ has a quiescent state where i is the trivial injection sending the single state of $\perp$ to the quiescent state of $\mathcal{B}$. Assertion 3 follows for $\preccurlyeq_i$ by assertion 1.
4. We show only the first relation, the second being rigorously symmetric. First, one has always $\mathcal{A} \trianglelefteq_{\pi_1} \mathcal{A} \times \mathcal{B}$ where $\pi_1 : S_{\mathcal{A}} \times S_{\mathcal{B}} \to S_{\mathcal{A}}$ is the projection over the first component. Second, if $\mathcal{B}$ has a quiescent state q, one has $\mathcal{A} \sqsubseteq_\iota \mathcal{A} \times \mathcal{B}$ where ι is the injection defined by $\iota(x) = (x, q)$ for all $x \in S_{\mathcal{A}}$ (the equality $\bar{\iota} \circ G_{\mathcal{A}} = (G_{\mathcal{A}} \times G_{\mathcal{B}}) \circ \bar{\iota}$ is indeed true over $S^{\mathbb{Z}}_{\mathcal{A}}$). If $\mathcal{B}$ has no quiescent state, just consider $\mathcal{B}^t$ and apply the previous reasoning to obtain:
$$\mathcal{A}^t \sqsubseteq \mathcal{A}^t \times \mathcal{B}^t = \big(\mathcal{A} \times \mathcal{B}\big)^t$$

and thus $\mathcal{A} \preccurlyeq_i \mathcal{A} \times \mathcal{B}$.

■

The 3 simulation quasi-orders are derived through bulking axiomatics from 3 different relations on local rules (see 2.1). There is *a priori* no reason why the differences between local relations should extend to differences between the 3 simulation quasi-orders. The following theorem shows that $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$ are nevertheless different.

Theorem 2.4. *The relations $\preccurlyeq_i$ and $\preccurlyeq_s$ are incomparable (no inclusion in either direction) and both strictly included in $\preccurlyeq_m$.*

Proof. We first show that there are CA $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \sqsubseteq \mathcal{B}$ but $\mathcal{A} \not\preccurlyeq_s \mathcal{B}$. Let $\mathcal{A} = \sigma \times \sigma^{-1}$ defined over states set $S_\mathcal{A} = \{0,1\} \times \{0,1\}$ and let $\mathcal{B}$ be the CA of radius 1 defined over $S_\mathcal{B} = S_\mathcal{A} \cup \{\#\}$ by:

$$f_\mathcal{B}(x,y,z) = \begin{cases} f_\mathcal{A}(x,y,z) & \text{if } x,y,z \in S_\mathcal{A}, \\ y & \text{else.} \end{cases}$$

One clearly has $\mathcal{A} \sqsubseteq_{Id} \mathcal{B}$. Now suppose $\mathcal{A} \preccurlyeq_s \mathcal{B}$. Without loss of generality we can assume that there are geometrical transforms $\alpha =< m, \tau, T, s >$ and $\beta =< m', 1, T', 0 >$ such that $\mathcal{A}^{\langle\alpha\rangle} \trianglelefteq_\pi \mathcal{B}^{\langle\beta\rangle}$. But, by definition of $\mathcal{B}$, there exist some state q_0 of $\mathcal{B}^{\langle\beta\rangle}$ (one can choose $\#^{m'}$) which is left invariant by iteration of $\mathcal{B}^{\langle\beta\rangle}$ whatever the context. Then $\pi(q_0)$ must be a state of $\mathcal{A}^{\langle\alpha\rangle}$ with the same property. This is impossible since either $s \neq T$ or $s \neq -T$ and thus some component of the future state of a cell of $\mathcal{A}^{\langle\alpha\rangle}$ is dependent of the state of a neighboring cell.

Now we show that there are $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \trianglelefteq \mathcal{B}$ but $\mathcal{A} \not\preccurlyeq_i \mathcal{B}$ and the theorem follows. Intuitively, $\mathcal{A}$ is a CA with two states, 0 and 1, whose behavior is to reduce ranges of 1's progressively until they reach size 1: at each time step the cells at each ends of a range of size 3 or more are turned into state 0 (only the right cell of range of size 2 is turned into 0). $\mathcal{B}$ has 3 states (0, 1 and 2) and has the following behavior: ranges of size 3 or more of non-zero states are reduced in a similar way by the two ends (states inside ranges are left unchanged), ranges of size 2 become an isolated 2 (left cell becomes 2 and right cell 0), and ranges of size 1 become an isolated 1. In a word, $\mathcal{B}$ reduce size of non-zero until size 1 but keeps the parity information at the end: an even range becomes eventually an isolated 2 and an odd range becomes an isolated 1 (see figure 1).

Formally, let $\pi : \{0,1,2\} \to \{0,1\}$ be the surjective function defined by $\pi(0) = 0$ and $\pi(x) = 1$ if $x \neq 0$. Now let $\mathcal{A}$ be the CA of radius 2 and states set $S_\mathcal{A} = \{0,1,2\}$ with local rule:

$$f_\mathcal{A}(x,y,z,t,u) = \begin{cases} 1 & \text{if } \overline{\pi}(xyztu) = 01110, \\ 2 & \text{if } \overline{\pi}(yztu) = 0110, \\ z & \text{if } \overline{\pi}(yzt) = 111 \text{ and if } \overline{\pi}(xyztu) \neq 01110, \\ z & \text{if } \overline{\pi}(yzt) = 010, \\ 0 & \text{in any other case.} \end{cases}$$

Finally, let $\mathcal{B}$ be the CA with states set $B = \{0,1\}$, radius 2 and local transition function

$$f_\mathcal{B}(x,y,z,t,u) = \begin{cases} 1 & \text{if } yzt = 111 \text{ or } yzt = 010 \text{ or } yztu = 0110, \\ 0 & \text{else.} \end{cases}$$

By construction, we have $\mathcal{A} \trianglelefteq_\pi \mathcal{B}$. Now suppose by contradiction that $\mathcal{A} \preccurlyeq_i \mathcal{B}$

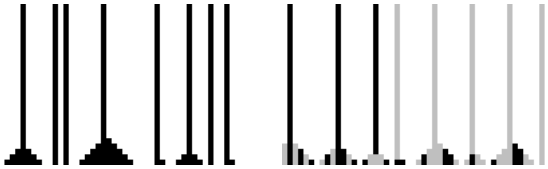


Figure 1: Behavior of $\mathcal{A}$ (left) and $\mathcal{B}$ (right). Time goes from bottom to top.

and more precisely:

$$\mathcal{A}^{\langle m',\tau',t',s'\rangle} \sqsubseteq_\phi \mathcal{B}^{\langle m,1,t,0\rangle},$$

where $\alpha =< m', \tau', t', s' >$ and $\beta =< m, 1, t, 0 >$ are suitable geometrical transforms. Let $u = 1^{m'}$ and $v = 0^{m'}$ (u and v are particular states of $\mathcal{A}^{\langle\alpha\rangle}$) and consider $U = \phi(u)$ and $V = \phi(v)$ (U and V belongs to $S_{\mathcal{B}}^{m'}$). Since configurations $\overline{u}$ and $\overline{v}$ are fixed points of $\mathcal{A}^{\langle\alpha\rangle}$, so are $\overline{U}$ and $\overline{V}$ for $\mathcal{B}^{\langle\beta\rangle}$. Moreover, one can check from the definition above that the state 0 is a 'blocking state' for $\mathcal{B}$: the half-configuration on the left of an occurrence of 0 evolve independently of the half-configuration on its right. So, if U contains one or more zero's, then any configuration of $\mathcal{B}^{\langle\beta\rangle}$ containing U^3 will contain at least one occurrence of U for ever (because it is the case for the configuration $\overline{U}$): this is in contradiction with the fact that the orbit of a configuration of the form ${}^\omega vuv^\omega$ do not contain any occurrence of u after one iteration of $\mathcal{A}^{\langle\alpha\rangle}$. Hence we have $\overline{\pi}(U) = 1^m$.

From this we deduce that $V = 0^m$ because configurations of the form ${}^\omega VU^nV^\omega$ are transformed into configurations where a single cell is not in state V (just consider the orbit of ${}^\omega vuv^\omega$ under $\mathcal{A}^{\langle\alpha\rangle}$) and large range of non-zero states are always turned into large range of zero's under $\mathcal{B}$.

Finally, we have $\overline{\pi}\big(\phi(0^{m'-1}1)\big) = 0^{m-1}1$ by considering the orbit of a configuration of the form ${}^\omega vu^2v^\omega$ under $\mathcal{A}^{\langle\alpha\rangle}$ and its counterpart of the form ${}^\omega VU^2V^\omega$ under $\mathcal{B}^{\langle\beta\rangle}$ (by the way, we also show that the shift parameter of transform α is 0). Now, letting $u' = 0^{m'-1}1$, we have on one hand the orbits of 2 configurations of the form ${}^\omega vu'u^{2n}v^\omega$ and ${}^\omega vu^{2n}v^\omega$ both leading to the same configuration of the form ${}^\omega vu'v^\omega$ under $\mathcal{A}^{\langle\alpha\rangle}$, and on the other hand, the orbits of ${}^\omega V\phi(u')U^{2n}V^\omega$ and ${}^\omega VU^{2n}V^\omega$ leading to different fixed points under $\mathcal{B}^{\langle\beta\rangle}$ due to different parity of non-zero ranges: this is a contradiction since $\overline{\phi} \circ \mathcal{A}^{\langle\alpha\rangle} = \mathcal{B}^{\langle\beta\rangle} \circ \overline{\phi}$. ■

3. Bottoms of the Orders

We have already seen (theorem 2.3) that $\perp$ is a global minimum for the 3 quasi-orders considered here. In this section, we study CA that are at the bottom of the quasi-orders. Formally, the only CA at level 0 is $\perp$ and a CA $\mathcal{A}$ is at level $n+1$ for a quasi-order $\preccurlyeq$ if:

1. $\mathcal{A}$ is not at level n and,
2. $\forall \mathcal{B} : \mathcal{B} \preccurlyeq \mathcal{A} \Rightarrow \mathcal{B} \in [\mathcal{A}]$ or $\mathcal{B}$ is at level i with $i \leq n$.

A cellular automaton is nilpotent if all initial configurations lead to the same configuration after a finite time. The property of nilpotency corresponds to a class at level 1 as shown by the following theorem.

Theorem 3.1. *Let $\preccurlyeq$ be a simulation relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Then the following CA are at level 1:*

1. *the set of nilpotent CA with 2 or more states, which is an equivalence class for $\sim$,*
2. *the set of CA which are periodic up to translation ($\mathcal{A}^t \circ \sigma_z = Id$) which is exactly the equivalence class for $\sim$ of the identity CA.*

Proof.

1. Nilpotency is equivalent to the existence of a uniform configuration reached in a fixed finite time from any configuration. This property of phase space is clearly invariant by geometrical transforms and preserved by taking sub-automata or quotient automata. So any nilpotent CA is at level at most 1. Moreover, for any nilpotent $\mathcal{A}$, there is t such that $\mathcal{A}^t$ is a constant function equal to some $\overline{q_a}$. If we consider any nilpotent $\mathcal{B}$ with at least 2 states, there is m such that $|S^m_{\mathcal{B}}| \geq |S_{\mathcal{A}}|$ and t' such that $\mathcal{B}^{t'}$ is a constant function equal to some $\overline{q_b}$. If we consider the geometrical transforms $\alpha =< 1, 1, t, 0 >$ and $\beta =< m, 1, t', 0 >$, then we have both $\mathcal{A}^{\langle\alpha\rangle} \sqsubseteq_i \mathcal{B}^{\langle\beta\rangle}$ and $\mathcal{A}^{\langle\alpha\rangle} \trianglelefteq_\pi \mathcal{B}^{\langle\beta\rangle}$ if i is such that $i(q_a) = q_b$ and π is such that $\pi(x) = q_a \iff x = q_b$.
2. Any CA which is periodic up to a translation is by definition equivalent to some identity CA and two identity CA with different state set are also clearly equivalent. The proof follow since the identity CA is at level 1. ■

The bottom of the quasi-order $\leqslant_\square$ was studied in [23]. The main result is the existence of an infinite familly of mutually incomparable CA at level 1: the familly of CA $\mathcal{Z}_p$ with p a prime number and where $\mathcal{Z}_p$ is a CA of radius 1 and state set $\{0, \ldots, p-1\}$ defined by the following local rule:

$$\delta_{\mathcal{Z}_p}(x, y, z) = x + y + z \bmod p.$$

There are strong connections between $\leqslant_\square$ and $\preccurlyeq_i$ and in fact the set of CA at level 1 are the same for these two quasi-orders.

Lemma 3.1. *If $\mathcal{A}$ is at level 1 for $\leqslant_\square$ then $\mathcal{A}$ is at level 1 for $\preccurlyeq_i$.*

Proof. If $\mathcal{B} \preccurlyeq_i \mathcal{A}$ then by lemma **??** there is some integer t and some transform β such that $\mathcal{B}^{\langle\beta\rangle} \sqsubseteq \mathcal{A}^{[t]}$. By theorem 2.3 we can suppose that $\mathcal{A}$ has radius 1 so $\mathcal{B}^{\langle\beta\rangle}$ has radius 1. Since $\mathcal{A}$ is at level 1 for $\leqslant_\square$, then either $\mathcal{B}^{\langle\beta\rangle} \in [\mathcal{A}]_\square$ or $\mathcal{B}^{\langle\beta\rangle} \in [\perp]_\square$. We deduce that either $\mathcal{B} \in [\mathcal{A}]_i$ or $\mathcal{B} \in [\perp]_i$. Hence $\mathcal{A}$ is at level at most 1 for $\preccurlyeq_i$ and it cannot be at level O since it is not in $[\perp]_i = [\perp]_\square$. ■

The previous lemma is not enough to show that the CA $\left(\mathcal{Z}_p\right)_p$ with p prime are mutually $\preccurlyeq_i$-incomparable because several equivalence classes for $\leqslant_\square$ can be included in a single class for $\preccurlyeq_i$. However we are going to show that this familly is a set of mutually incomparable CA for all quasi-orders considered in this paper[2]. Moreover, for $\leqslant_\square$ and $\preccurlyeq_i$, they are all at level 1. The proof relies on the following result already used for the case of $\leqslant_\square$.

A CA (S, r, f) is *LR-permutative* if the two following functions are bijections for all $a_1, \ldots, a_{2r}$:

[2]The proof we give here was suggested by E. Jeandel.

- $x \mapsto f(a_1, \ldots, a_{2r}, x)$ and
- $x \mapsto f(x, a_1, \ldots, a_{2r})$.

Theorem 3.2 ([21]). *Let p be a prime number and $t \geq 1$. Then we have:*

1. ${\mathcal{Z}_p}^{[t]}$ *is LR-permutative;*
2. *if $\mathcal{A} \sqsubseteq {\mathcal{Z}_p}^{[t]}$ then p divides $|S_\mathcal{A}|$.*

To take into account use of $\trianglelefteq$ in simulation we will use the following lemma.

Lemma 3.2. *If $\mathcal{B}$ is LR-permutative and $\mathcal{A} \trianglelefteq \mathcal{B}$ then $|S_\mathcal{A}|$ divides $|S_\mathcal{B}|$.*

Proof. To simplify notations, we suppose that $\mathcal{B}$ is of radius 1 (the proof works the same way for higher radii). Suppose $\mathcal{A} \trianglelefteq_\pi \mathcal{B}$. By surjectivity of π, it is sufficient to show that π is balanced, *i.e.* such that for all $x, y \in S_\mathcal{A}$:

$$|\{e : \pi(e) = x\}| = |\{e : \pi(e) = y\}|.$$

Consider any $x, y \in S_\mathcal{A}$. Let $a, b \in S_\mathcal{B}$ be such that $\pi(a) = x$ and $\pi(b) = y$ and consider any $c \in S_\mathcal{B}$. By R-permutativity there is $d \in S_\mathcal{B}$ such that $f_\mathcal{B}(a, c, d) = b$. Now for any $a' \in S_\mathcal{B}$ such that $\pi(a') = \pi(a)$, we must have $\pi\big(f_\mathcal{B}(a', c, d)\big) = \pi(b)$ because $\mathcal{A} \trianglelefteq_\pi \mathcal{B}$. Moreover, by L-permutativity, all the images $f_\mathcal{B}(a', c, d)$ are different which proves:

$$|\{a : \pi(a) = x\}| \leq |\{b : \pi(b) = y\}|.$$

The balance of π follows by symmetry. ■

The results above are the key ingredient of the following theorem.

Theorem 3.3. *Let $\preccurlyeq$ be any relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Let p and q be two distinct prime numbers. Then we have:*

1. $\mathcal{Z}_p \not\preccurlyeq \mathcal{Z}_q$
2. $\mathcal{Z}_p$ *is at level* 1 *for* $\preccurlyeq_i$*;*

Proof. 2 follows immedialty from lemma 3.1 and the fact that $\mathcal{Z}_p$ is at level 1 for $\leqslant_\square$ (corollary 2 of [21]). To prove 1 it is enough to prove $\mathcal{Z}_p \not\preccurlyeq_m \mathcal{Z}_q$. Suppose by contradiction that $\mathcal{Z}_p \preccurlyeq_m \mathcal{Z}_q$, or equivalently by lemma **??**, that there are a CA $\mathcal{A}$, a transform α and an integer t such that ${\mathcal{Z}_p}^{\langle\alpha\rangle} \trianglelefteq \mathcal{A} \sqsubseteq {\mathcal{Z}_q}^{[t]}$. Then, combining lemma 3.2 and theorem 3.2, we deduce that the number of states of ${\mathcal{Z}_p}^{\langle\alpha\rangle}$ is a power of q which contradicts the fact that p and q are two distinct primes. ■

We will now study a familly of CA which shows that there are infinitely many incomparable CA at any finite level greater than 3 for any of the 3 simulation quasi-orders of the paper.

We denote by $\sigma_{n,z}$ the translation CA with n states $\{1, \ldots, n\}$ and translation vector z defined by:

$$\sigma_{n,z}(c)_{z'} = c_{z'-z}.$$

We then consider cartesian products of such CA. Since $\sigma_{n,z} \times \sigma_{p,z} \equiv \sigma_{np,z}$, we can focus on consider cartesian products where all vectors are distinct.

The next lemma shows that the structure of product of translation is preserved when taking sub-automata and quotient-automata.

Lemma 3.3. *Let $\mathcal{B} = \prod_{i=1}^{p} \sigma_{n_i,z_i}$ (with z_i all distinct) and suppose $\mathcal{A}$ is such that $\mathcal{A} \trianglelefteq\!\!\sqsubseteq \mathcal{B}$ then $\mathcal{A}$ is isomorphic to $\prod_{j=1}^{k} \sigma_{n'_{i_j},z_{i_j}}$ where $1 \leq i_j \leq n$ and $2 \leq n'_{i_j} \leq n_{i_j}$.*

Proof. The lemma is straightforward if we replace $\trianglelefteq\!\!\sqsubseteq$ by $\sqsubseteq$. So it is enough to show that it is also true when replacing $\trianglelefteq\!\!\sqsubseteq$ by $\trianglelefteq$. Suppose that $\mathcal{A} \trianglelefteq_h \mathcal{B}$. Let i be fixed between 1 and p. Consider any $x, x' \in \{1, \ldots, n_i\}$ and any $q, q' \in S_{\mathcal{B}}$ such that $\pi_i(q) = \pi_i(q') = x$ (where π_i denotes the projection on the ith coordinate). Denote by q_+ and q'_+ the states obtained from q and q' by changing the ith coordinate to x'. Then we have:

$$h(q) = h(q') \Rightarrow h(q_+) = h(q'_+)$$

because one can build two configurations c and c' of $\mathcal{B}$ such that $c' = \mathcal{B}(c)$ and $c(0) = q$ and $c'(-z_i) = q'$, and obtain c'_+ and c' by changing q to q_+ at $c(0)$ and q' to q'_+ at $c'(-z_i)$. Hence, if there exist two states q and q' with the same image by f and which agree on all components except component i where q equals x and q' equals x', then x and x' in ith component can always be exchanged without afecting the image by h whatever the content of other components. Therefore, taking the apropriate quotient automaton $\mathcal{C}$, we have $\mathcal{A} \trianglelefteq \mathcal{C} \trianglelefteq \mathcal{B}$ where $\mathcal{C}$ equals $\mathcal{B}$ but with 1 state less in ith component. Applying this reasoning inductively, we finally have $\mathcal{A} \trianglelefteq_g \mathcal{C}_0 \trianglelefteq \mathcal{B}$ where $\mathcal{C}_0$ is of the form $\prod_{j=1}^{k} \sigma_{n'_{i_j},z_{i_j}}$ where $1 \leq i_j \leq n$ and $2 \leq n'_{i_j} \leq n_{i_j}$ (component reduced to 1 state during the induction can be eliminated) and g is such that changing the value of any component will change the image by g. Now suppose by contradiction that g is not injective. Then there are states q and q' of $\mathcal{C}_0$ such that $g(q) = g(q')$. Let $c = \overline{q}$ and c' be equal to c except on position 0 where it is in state q'. By hypothesis, at any position z, $G_{\mathcal{C}_0}(c)$ and $G_{\mathcal{C}_0}(c')$ must be in states having the same image by g. But since $\mathcal{C}_0$ is a product of translations with distinct vectors, there must be some position z where $G_{\mathcal{C}_0}(c)$ and $G_{\mathcal{C}_0}(c')$ are in states which differ on one component only: this is in contradiction with the hypothesis on g. Hence g is injective and therefore $\mathcal{A} \equiv \mathcal{C}_0$. ■

If $\mathcal{A}$ is a proudct of translations with vectors $z_1 < \ldots < z_a$, we denote by $\chi(\mathcal{A})$ the following characteristic sequence (provided $a \geq 3$):

$$\chi(\mathcal{A}) = \left(\frac{z_3 - z_1}{z_2 - z_1}, \ldots, \frac{z_a - z_1}{z_2 - z_1} \right).$$

The usefulness of this notion of characterisitic sequence is revealed by the following theorem.

Theorem 3.4. *Let $\preccurlyeq$ be a relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Let $\mathcal{A}$ be a product of $a \geq 3$ translations with distinct vectors and with characterisitic sequence $\chi(\mathcal{A}) = (\alpha_1, \ldots, \alpha_{a-2})$. If $\mathcal{B} \preccurlyeq \mathcal{A}$ then $\mathcal{B}$ is equivalent to some $\mathcal{C}$ which is a product of a subset of b translations of $\mathcal{A}$. Moreover, we have the following properties:*

1. *if $b = a$ then $\mathcal{C}$ has the same characteristic sequence than $\mathcal{A}$;*
2. *if $b = a - 1$ and $b \geq 3$ then the characteristic sequence of $\mathcal{C}$ has one of the following form:*

- $(\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_{a-2})$
- $\left(\frac{\alpha_2}{\alpha_1}, \ldots, \frac{\alpha_{a-2}}{\alpha_1}\right)$
- $\left(\frac{\alpha_2-1}{\alpha_1-1}, \ldots, \frac{\alpha_{a-2}-1}{\alpha_1-1}\right)$

3. *if $\mathcal{C}$ has not $\chi(\mathcal{A})$ for characteristic sequence then $\mathcal{A} \not\preccurlyeq \mathcal{C}$.*

PROOF. Let $z_1 < z_2 < \ldots < z_a$ be the ordered list of translation vectors of $\mathcal{A}$. Since $\mathcal{B} \preccurlyeq \mathcal{A}$, there is some $\mathcal{C}$ equivalent to $\mathcal{B}$ and some integer $t \geq 1$ such that $\mathcal{C} \trianglelefteq\!\sqsubseteq \mathcal{A}^{[t]}$ (by lemma **??**). We deuduce from lemma 3.3 that $\mathcal{C}$ is isomorphic to a product of translations whose vectors are a subset of the familly (z_i) since $\mathcal{A}$ and $\mathcal{A}^{[t]}$ have identical translation vectors, $\mathcal{C}$ must have the same characteristic sequence than $\mathcal{A}$ if it has the same number of translation vectors. When $b = a - 1$ and $b \geq 3$, it is straightforward to check that the 3 possible forms of the characteristic sequence of $\mathcal{C}$ correspond to the case where the missing vector is z_i, z_2 and z_1 respectively. To prove the last assertion of the theorem, it is sufficient to check that for any transform α of the form $< m, 1, mt, mz >$ (we can restrict to such transforms by lemma **??**, $\mathcal{C}$ and $\mathcal{C}^{\langle\alpha\rangle}$ are products of translations with the same characteristic sequence because each vector z_i of $\mathcal{C}$ becomes $tz_i + z$ in $\mathcal{C}^{\langle\alpha\rangle}$. ■

Notice that the previous theorem implies that there are CA $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ which are all equivalent but such that $\mathcal{A} \times \mathcal{A}$ is not equivalent to $\mathcal{B} \times \mathcal{C}$.

The next lemma give canonical members of their equivalence classes.

Lemma 3.4. *Let $\sim$ be the equivalence relation induced by any of the quasi-order $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Consider any $t \neq 0$, any z and any product of translations of the form $\mathcal{A} = \prod_{i=1}^{p} \sigma_{n_i, tz_i+z}$, . Then we have:*

$$\mathcal{A} \sim \prod_{1 \leq i \leq p} \sigma_{2,z_i}$$

PROOF. Let $\mathcal{B} = \prod_{i=1}^{p} \sigma_{2,z_i}$ and let $m = \max n_i$. It is straightforward to check that one has $\mathcal{A} \trianglelefteq \mathcal{B}^{\langle m,1,mt,mz\rangle}$ and $\mathcal{A} \sqsubseteq \mathcal{B}^{\langle m,1,mt,mz\rangle}$ on one hand, and, on the other hand, $\mathcal{B}^{\langle 1,1,t,z\rangle} \trianglelefteq \mathcal{A}$ and $\mathcal{B}^{\langle 1,1,t,z\rangle} \sqsubseteq \mathcal{A}$. ■

Theorem 3.4 and lemma 3.4 give a complete characterisation of the position of products of shift in the quasi-orders considered in this paper. We will use it later in section 4.1 but we now state the main result of this section concerning levels at the bottom of the quasi-orders.

Corollary 3.1. *Let $\preccurlyeq$ be a relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. For any $n \geq 3$, there are infinitely many incomparable CA at level n for $\preccurlyeq$.*

PROOF. We have shown in theorem 3.1 that translation CA are at level 1. Lemma 3.3 together with lemma 3.4 show that a product of two translations (with distinct vectors) is at level 2. By theorem 3.4 we conclude that any product of n translations with distinct vectors is at level n and two such CA are incomparable if they have different characteristic sequences provided $n \geq 3$. ■

4. Structural properties

In this section we study in various ways the order structure induced by the simulation relations defined above.

4.1. Cartesian Products and Lack of (Semi-)Lattice Structure

The next theorem shows how some simulations by Cartesian products of CA can be transposed to components of the product.

Theorem 4.1. *Let $\mathcal{A}$ be a CA with 2 states and let $\preccurlyeq$ be a simulation relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. For any $\mathcal{B}$ and $\mathcal{C}$, if $\mathcal{B} \times \mathcal{C}$ strongly $\preccurlyeq$-simulates $\mathcal{A}$ then either $\mathcal{A} \preccurlyeq \mathcal{B}$ or $\mathcal{A} \preccurlyeq \mathcal{C}$.*

Proof. Let $S_{\mathcal{A}} = \{a_1, a_2\}$. First, we show that $\mathcal{A} \sqsubseteq_\iota \mathcal{B} \times \mathcal{C}$ implies either $\mathcal{A} \sqsubseteq \mathcal{B}$ or $\mathcal{A} \sqsubseteq \mathcal{C}$ which is sufficient to prove the theorem for $\preccurlyeq_i$ and $\preccurlyeq_m$. Since $\iota(a_1) \neq \iota(a_2)$ we have either $\pi_1(i(a_1)) \neq \pi_1(i(a_2))$ or $\pi_2(i(a_1)) \neq \pi_2(i(a_2))$ where π_1 and π_2 are projection over first and second component respectively. We suppose the first case (the second is symmetric) and so $\pi_1 \circ i : S_{\mathcal{A}} \to S_{\mathcal{B}}$ is injective. Moreover, since

$$\overline{\pi_1 \circ \iota} \circ G_{\mathcal{A}} = \overline{\pi_1} \circ G_{\mathcal{B}\times\mathcal{C}} \circ \overline{\iota} = G_{\mathcal{B}} \circ \overline{\pi_1 \circ \iota},$$

we conclude that $\mathcal{A} \sqsubseteq_{\pi_1 \circ \iota} \mathcal{B}$.

Second, we show that $\mathcal{A} \trianglelefteq_s \mathcal{B} \times \mathcal{C}$ implies either $\mathcal{A} \preccurlyeq_s \mathcal{B}$ or $\mathcal{A} \preccurlyeq_s \mathcal{C}$ which is sufficient to prove the theorem for $\preccurlyeq_s$. Let $I_{\mathcal{A}}$ be the set of states that can be reached after one step of $\mathcal{A}$ (formally, $I_{\mathcal{A}} = f_{\mathcal{A}}(S_{\mathcal{A}}, \dots, S_{\mathcal{A}})$) and $I_{\mathcal{B}}$ and $I_{\mathcal{C}}$ be similar sets for $\mathcal{B}$ and $\mathcal{C}$.

- We first suppose that $\mathcal{B}$ and $\mathcal{C}$ are such that each uniform configuration is either a fixed-point or without any uniform antecedent. If $a_1 \notin I_{\mathcal{A}}$ then for any b and c such that $s(b, c) = a_1$ we have either $b \notin I_{\mathcal{B}}$ or $c \notin I_{\mathcal{C}}$. We suppose the first case (the second is analogous) and then we have $\mathcal{A} \trianglelefteq_\zeta \mathcal{B}$ where $\zeta : S_{\mathcal{B}} \to S_{\mathcal{A}}$ is defined by $\zeta(b) = a_1$ and $\zeta(x) = a_2$ for $x \neq b$.

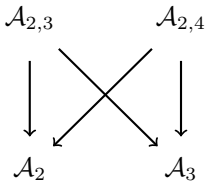
  If $a_2 \notin I_{\mathcal{A}}$ we apply the same reasoning and so we are left with the case $I_{\mathcal{A}} = S_{\mathcal{A}}$. Since pairs of $S_{\mathcal{B}} \times S_{\mathcal{C}}$ are 2-colored via s, there must be two pairs of different colors which agree on a component. Suppose it is the first component (the other case is symmetric), we have $b_1, b_2 \in S_{\mathcal{B}}$ and $c \in S_{\mathcal{C}}$ such that $s(b_1, c) = a_1$ and $s(b_2, c) = a_2$. Consider the set $X = \{(b_1, c), (b_2, c)\}$. Since $\overline{s} \circ (G_{\mathcal{B}} \times G_{\mathcal{C}}) = G_{\mathcal{A}} \circ \overline{s}$ and $\overline{s}\big(X^{\mathbb{Z}}\big) = S_{\mathcal{A}}^{\mathbb{Z}}$ and $I_{\mathcal{A}} = S_{\mathcal{A}}$ we necessarily have $s(S_{\mathcal{B}}, d) = S_{\mathcal{A}}$ where d defined by $\overline{d} = G_{\mathcal{C}}(\overline{c})$. d is quiescent by hypothesis on $\mathcal{C}$. So we have $\mathcal{A} \trianglelefteq_\zeta \mathcal{B}$ with $\zeta : S_{\mathcal{B}} \to S_{\mathcal{A}}$ defined by $\zeta(x) = s(x, d)$ (ζ is onto by choice of d).

- Now suppose that the hypothesis on $\mathcal{B}$ and $\mathcal{C}$ are not fulfilled. Then, if $t = |S_{\mathcal{B}}|! \times |S_{\mathcal{C}}|!$, both $\mathcal{B}^t$ and $\mathcal{C}^t$ are guarantied to fulfill the required hypothesis (because any uniform configuration is either in a cycle of uniform configurations, or without uniform antecedent arbitrarily far in the past). Since $\mathcal{A}^t \trianglelefteq \big(\mathcal{B} \times \mathcal{C}\big)^t = \mathcal{B}^t \times \mathcal{C}^t$, it suffices to apply the previous reasoning on $\mathcal{A}^t$, $\mathcal{B}^t$ and $\mathcal{C}^t$ to conclude either $\mathcal{A}^t \trianglelefteq \mathcal{B}^t$ or $\mathcal{A}^t \trianglelefteq \mathcal{C}^t$. In either case the theorem follows. ■

Despite the properties of Cartesian product with respect to simulation quasi-orders (theorem 4.1 and assertion 4 of theorem 2.3), this natural operation is not a supremum in any of the quasi-order. In fact these quasi-orders don't admit any supremum operation as shown by the theorem below. The proof relies on the study of products of translation CA.

Theorem 4.2. *Let $\preccurlyeq$ be a relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. Then the ordered structure $\left(AC/\sim, \preccurlyeq\right)$ is neither an upper semi-lattice, nor a lower semi-lattice.*

Proof. Let $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_{2,3}$ and $\mathcal{A}_{2,4}$ be products of translations with characteristic sequences (2), (3), $(2,3)$ and $(2,4)$ respectively. Theorem 3.4 and lemma 3.4 show that they induce the following structure in $\preccurlyeq$:



where an arrows from $\mathcal{A}$ to $\mathcal{B}$ means $\mathcal{B} \preccurlyeq \mathcal{A}$ and if $\mathcal{B} \preccurlyeq \mathcal{C} \preccurlyeq \mathcal{A}$ then either $\mathcal{B} \sim \mathcal{C}$ or $\mathcal{C} \sim \mathcal{A}$. This shows that the pair $\mathcal{A}_2$, $\mathcal{A}_3$ has no supremum and that the pair $\mathcal{A}_{2,3}$, $\mathcal{A}_{2,4}$ has no infinimum. ■

4.2. Ideals and Filters

Although the structures $\left(AC, \preccurlyeq\right)$ studied in this paper are not semi-lattices (see above), many classical properties of cellular automata are nicely captured through *ideals* and *filters*. Well-known in lattice theory and algebra, the notions of ideal and filter can also by defined for an arbitrary (quasi-)ordered structure [6]. For the structure $\left(AC, \preccurlyeq\right)$, an ideal I is a set of CA such that:

- if $\mathcal{A} \in I$ and $\mathcal{B} \preccurlyeq \mathcal{A}$ then $\mathcal{B} \in I$;
- for any $\mathcal{A}, \mathcal{B} \in I$ there is some $\mathcal{C} \in I$ such that $\mathcal{A} \preccurlyeq \mathcal{C}$ and $\mathcal{B} \preccurlyeq \mathcal{C}$.

Moreover, I is said *principal* if there is some $\mathcal{A}_I$ such that $\mathcal{A} \in I \iff \mathcal{A} \preccurlyeq \mathcal{A}_I$. The notion of filter and principal filter are dual (replacing all $X \preccurlyeq Y$ by $Y \preccurlyeq X$).

Given a set I of CA, the 3 following conditions are sufficient for I to be an ideal for the simulation $\preccurlyeq_i$ (resp. $\preccurlyeq_s$, or $\preccurlyeq_m$):

1. $\mathcal{A} \in I \iff \mathcal{A}^{\langle\alpha\rangle} \in I$ for any transform α,
2. if $\mathcal{B} \in I$ and $\mathcal{A} \sqsubseteq \mathcal{B}$ (resp. $\mathcal{A} \trianglelefteq \mathcal{B}$, or $\mathcal{A} \trianglelefteq\!\!\!\sqsubseteq \mathcal{B}$) then $\mathcal{A} \in I$,
3. if $\mathcal{A} \in I$ and $\mathcal{B} \in I$ then $\mathcal{A} \times \mathcal{B} \in I$.

Most of the proofs below follow this scheme. The following theorem shows that several dynamical properties of global rules of CA correspond to ideals in the quasi-orders. A CA is nilpotent over periodic configurations if there exists a spatially periodic configuration c_0 such that all spatially periodic configurations lead in finite time to c_0.

Theorem 4.3. *Let $\preccurlyeq$ be a simulation relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. For each property $\mathcal{P}$ in the following list, the set of CA having property $\mathcal{P}$ is an ideal of $\left(AC, \preccurlyeq\right)$:*

- *surjectivity,*
- *reversibility,*
- *nilpotency over periodic configurations.*

PROOF. First, from the point of view of global maps, a geometric transform consists in iterating or composing with bijective maps. So the properties of being surjective or reversible are left unchanged by geometrical transforms. Besides, geometrical transforms send periodic configurations to periodic configurations, temporal cycles of configuration to temporal cycles (eventually reduced to a single configuration), and attraction basin of such cycles to attraction basin of cycles. Hence, nilpotency over periodic configuration, which is equivalent to the existence of a temporal cycle having all periodic configurations in its attraction basin, is preserved by geometrical transforms. By similar reasoning on the phase space, it is straightforward to check that $\mathcal{A}$ is nilpotent over periodic configurations if $\mathcal{B}$ is and $\mathcal{A} \sqsubseteq \mathcal{B}$ or $\mathcal{A} \trianglelefteq \mathcal{B}$. And $\mathcal{A} \times \mathcal{B}$ is nilpotent over periodic configurations if both $\mathcal{A}$ and $\mathcal{B}$ are. So nilpotency over periodic configurations induces an ideal for $\preccurlyeq$.

It is also clear that surjectivity and reversibility are preserved by cartesian product. Now suppose $\mathcal{A} \trianglelefteq_\pi \mathcal{B}$. If $\mathcal{B}$ is surjective then so is $\mathcal{A}$ since $G_\mathcal{A} \circ \overline{\pi} = \overline{\pi} \circ G_\mathcal{B}$ and π is by definition surjective. If $\mathcal{B}$ is reversible, consider any map ϕ such that $\pi \circ \phi = Id$ and let $\mathcal{A}_{-1}$ be the CA over state set $S_\mathcal{A}$ and defined by the global map $G = \overline{\pi} \circ G_\mathcal{B}^{-1} \circ \overline{\phi}$ (it is a shift-commuting continuous map). Since $G_\mathcal{A} \circ \overline{\pi} = \overline{\pi} \circ G_\mathcal{B}$, one can check that $G_\mathcal{A} \circ G = Id$ so $\mathcal{A}$ is reversible.

Finally, suppose $\mathcal{A} \sqsubseteq_\iota \mathcal{B}$. If $\mathcal{B}$ is reversible then $\mathcal{A}$ is also reversible since $\overline{\iota} \circ G_\mathcal{A} = G_\mathcal{B} \circ \overline{\iota}$ and ι is by definition injective. If $\mathcal{B}$ is surjective, then so is $\mathcal{A}$ because $\mathcal{B}$ being injective over finite configurations (Moore-Myhill theorem[3]) $\mathcal{A}$ is also injective over finite configurations (ι maps finite configurations to finite configurations). ■

Theorem 4.4. *Let $\mathcal{A}$ and $\mathcal{B}$ be two reversible CA and $\preccurlyeq$ be a simulation relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. If $\mathcal{A} \preccurlyeq \mathcal{B}$ then $\mathcal{A}^{-1} \preccurlyeq \mathcal{B}^{-1}$.*

PROOF. First, it is straightforward to check that the inverse of geometrically transformed instances of $\mathcal{A}$ are transformed instances of the inverse of $\mathcal{A}$. Using what was shown above concerning reversibility, it is thus sufficient to prove the 2 following properties:

- $\mathcal{A} \sqsubseteq_\iota \mathcal{B}$ implies $\mathcal{A}^{-1} \sqsubseteq_\iota \mathcal{A}^{-1}$,
- $\mathcal{A} \trianglelefteq_g \mathcal{B}$ implies $\mathcal{A}^{-1} \trianglelefteq_g \mathcal{A}^{-1}$.

In the first case we have:

$$G_\mathcal{B} \circ \overline{\iota} = \overline{\iota} \circ G_\mathcal{A} \Rightarrow \overline{\iota} = G_\mathcal{B}^{-1} \circ \overline{\iota} \circ G_\mathcal{A} \Rightarrow \overline{\iota} \circ G_\mathcal{A}^{-1} = G_\mathcal{B}^{-1} \circ \overline{\iota}$$

[3]In [13], one can find the following theorem: a CA is surjective if and only if there is no pair of finite configurations (*i.e.* uniform except on a finite region) having the same image. The original formulation of the Moore-Myhill theorem [24, 25] supposes the existence of a quiescent state.

each equality being true on $S^{\mathbb{Z}}_{\mathcal{A}}$. In the second case we have:

$$G_{\mathcal{A}} \circ \overline{g} = \overline{g} \circ G_{\mathcal{B}} \Rightarrow G_{\mathcal{A}} \circ \overline{g} \circ G_{\mathcal{B}}^{-1} = \overline{g} \Rightarrow \overline{g} \circ G_{\mathcal{B}}^{-1} = G_{\mathcal{A}}^{-1} \circ \overline{g}$$

each equality being true on $S^{\mathbb{Z}}_{\mathcal{B}}$. ■

Theorem 4.5. *Let $\preccurlyeq$ be $\preccurlyeq_i$ or $\preccurlyeq_m$. Then the ideal of reversible CA is principal: there is a reversible CA $\mathcal{A}$ such that*

$$\mathcal{B} \text{ reversible} \iff \mathcal{B} \preccurlyeq \mathcal{A}.$$

Proof. In [8], a reversible CA $\mathcal{B}$ able to simulate any reversible CA is constructed. The notion of simulation used is included in $\preccurlyeq_i$ and therefore in $\preccurlyeq_m$. The implication $\Rightarrow$ is thus proven and the converse implication is proven by theorem 4.3. ■

For the ideal of surjective CA, the principality is still an open problem in dimension 1.

Open Problem 1. *Is the ideal of surjective CA principal, and for which simulation quasi-order?*

Limit sets of CA has received a lot of attention in the literature [5, 15, 11]. The limit set of $\mathcal{A}$ is the set $\Omega_{\mathcal{A}}$ of configurations having predecessors arbitrarily far in the past, formally:

$$\Omega_{\mathcal{A}} = \bigcap_t G^t_{\mathcal{A}}\big(S^{\mathbb{Z}}_{\mathcal{A}}\big).$$

The next theorem shows that the class of CA with a sofic limit set is nicely captured by $\preccurlyeq_s$.

Theorem 4.6. *The set of CA with a sofic limit set is an ideal for $\preccurlyeq_s$.*

Proof. For CA of dimension 1, having a sofic limit set is equivalent to having a regular limit language [35]. It is clear that this latter property is left unchanged by geometrical transforms (the limit language is not affect by iterations and shifts, the regularity of the language is not affected by packing). Hence, it is sufficient to show that if $\mathcal{B}$ has a regular limit language and $\mathcal{A} \trianglelefteq_g \mathcal{B}$ then $\mathcal{A}$ also has a regular limit language. Since regular languages are closed by substitution (a classical result which can be found in [14]), it is sufficient to prove that $\Omega_{\mathcal{A}} = \overline{g}\big(\Omega_{\mathcal{B}}\big)$. This last assertion is a direct consequence of $\mathcal{A} \trianglelefteq_g \mathcal{B}$, since the following equality holds by recurrence on t:

$$\overline{g}\big(G^t_{\mathcal{B}}(S^{\mathbb{Z}}_{\mathcal{B}})\big) = G^t_{\mathcal{A}}(S^{\mathbb{Z}}_{\mathcal{A}}).$$

■

The properties considered above are purely dynamic: they can be expressed as structural properties of the phase space with the reachability relation only. We now consider properties from topological dynamics: they are expressed with both the reachability relation and the toppology (Cantor distance) of the space of configurations.

The properties we will consider rely on the equicontinuity classification of P. Kůrka [18]. Let $\mathcal{A}$ be any CA of states set Q and global rule G and denote by d the Cantor distance over $Q^{\mathbb{Z}}$.

- $x \in Q^{\mathbb{Z}}$ is an *equicontinuity point* for $\mathcal{A}$ if

$$\forall \epsilon, \exists \delta, \forall y \in Q^{\mathbb{Z}} : d(x,y) \leq \delta \Rightarrow \forall t, d(G^t(x), G^t(y)) \leq \epsilon.$$

- $\mathcal{A}$ is *sensitive to initial conditions* if

$$\exists \epsilon, \forall \delta, \forall x \in Q^{\mathbb{Z}} \exists y \in Q^{\mathbb{Z}} \exists t : d(x,y) \leq \delta \text{ and } d\big(G^t(x), G^t(y)\big) \geq \epsilon.$$

- $\mathcal{A}$ is *(positively) expansive* if

$$\exists \epsilon, \forall x, y \in Q^{\mathbb{Z}} : x = y \iff \forall t, d\big(G^t(x), G^t(y)\big) \leq \epsilon.$$

The classification of P. Kůrka is the following:

K_1 is the set of CA for which all configurations are equicontinuity points,

K_2 is the set of CA having equicontinuity points,

K_3 is the set of CA sensitive to initial conditions,

K_4 is the set of expansive CA.

The weakness of this classification is its lack of shift-invariance: the identity and the elementary translation belong to different classes (K_1 and K_3 respectively). Several attempts have been made to overcome this problem by changing the topology [4]. More recently, a new approach has been proposed [30]: the Cantor topology is conserved (with all its good properties) but the topological properties are enriched with a new parameter (a velocity) which is used as the reference direction of information propagation in space-time. The original definitions of P. Kůrka are thus obtained by choosing velocity 0, but now identity and elementary translations are assigned to the same class (with different velocities). This directionnal dynamic approach is more suitable for our study since, by definition, the equivalence classes of any of our quasi-order is shift-invariant. We will define 4 classes based on the existence of some direction for which some dynamical behaviour is observed.

We say that $\mathcal{A}$ is a *rescaling* of $\mathcal{B}$ if there are transforms α and β such that $\mathcal{A}^{\langle\alpha\rangle} \equiv \mathcal{B}^{\langle\beta\rangle}$. We then consider the following 4 classes:

- the set $\mathcal{T}_1$ of CA which are a rescaling of some equicontinuous CA,
- the set $\mathcal{T}_2$ of CA which are a rescaling of some CA having equicontinuity points,
- the set $\mathcal{T}_3$ of CA which are not in $\mathcal{T}_2$, *i.e.* CA which are sensitive in every directions[4],
- the set $\mathcal{T}_4$ of CA which are a rescaling of some (positively) expansive CA.

Figure 2 is justified by the following theorem.

[4]For one-dimensional CA, the set of sensitive CA is the complement of the set of CA having equicontinuity points (see [18]). In [30], this complementarity is shown for any direction.

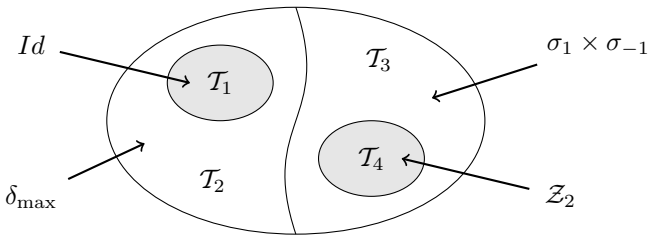


Figure 2: Four kinds of topological dynamics.

Theorem 4.7. *We have the following inclusions:*

1. $\mathcal{T}_1 \subseteq \mathcal{T}_2$,
2. $\mathcal{T}_4 \subseteq \mathcal{T}_3$.

Moreover, each of the set $\mathcal{T}_1$, $\mathcal{T}_2 \setminus \mathcal{T}_1$, $\mathcal{T}_3 \setminus \mathcal{T}_4$ and $\mathcal{T}_4$ is non-empty.

Proof. The first inclusion follows from definitions. The second follows from proposition 3.2 of [30] which asserts that the set of direction with equicontinuity points and the set of expansive directions cannot be simultaneously empty.

Non-emptyness of $\mathcal{T}_1$ and $\mathcal{T}_4$ follow from the existence of equicontinuous (*e.g.* the identity) and (positively) expansice CA (*e.g.* $\mathcal{Z}_2$). Moreover, any CA having an equicontinuity point which is not equicontinuous (*e.g.* the CA of local rule $\delta_{\max}(a,b,c) = \max(a,b,c)$) cannot be in $\mathcal{T}_1$ (equicontinuity is preserved by rescaling), so it is in $\mathcal{T}_2 \setminus \mathcal{T}_1$. Finally, by proposition 3.2 of [30], $\sigma_1 \times \sigma_{-1} \in \mathcal{T}_3 \setminus \mathcal{T}_4$ because any direction is either a direction of right-expansivity, or a direction of left-expansivity, neither both. ■

Theorem 4.8.

1. *$\mathcal{T}_1$ is an ideal for any simulation $\preccurlyeq$ among $\preccurlyeq_s$, $\preccurlyeq_i$ and $\preccurlyeq_m$;*
2. *$\mathcal{T}_2$ is an ideal for $\preccurlyeq_s$;*
3. *$\mathcal{T}_4$ is an ideal for $\preccurlyeq_i$.*

Proof. First, consider any $\mathcal{A}, \mathcal{B} \in \mathcal{T}_1$. Then there are CA $\mathcal{A}'$ and $\mathcal{B}'$ which are both equicontinuous and $\preccurlyeq$-equivalent to $\mathcal{A}$ and $\mathcal{B}$ respectively. Then, if $\mathcal{C} = \mathcal{A}' \times \mathcal{B}'$ we have $\mathcal{C} \in \mathcal{T}_1$ and by theorem 2.3 we have both $\mathcal{A} \preccurlyeq \mathcal{C}$ and $\mathcal{B} \preccurlyeq \mathcal{C}$. The same reasoning can be applied to $\mathcal{T}_4$ and $\mathcal{T}_2$. Thus we have shown the second condition of the definition of ideals for the 3 properties considered here.

To conclude the theorem, and since the 3 properties considered are by definition invariant by rescaling, it is sufficient to prove:

- if $\mathcal{A} \sqsubseteq \mathcal{B}$ or $\mathcal{A} \trianglelefteq \mathcal{B}$ then $\mathcal{B}$ equicontinuous $\Rightarrow$ $\mathcal{A}$ equicontinuous;
- if $\mathcal{A} \trianglelefteq \mathcal{B}$ then $\mathcal{B}$ has equicontinuous points $\Rightarrow$ $\mathcal{A}$ has equicontinuous points;
- if $\mathcal{A} \sqsubseteq \mathcal{B}$ then $\mathcal{B}$ expansive $\Rightarrow$ $\mathcal{A}$ expansive.

The first assertion follows from the characterisation of equicontinuous CA as ultimately periodic CA [18].

For the second assertion, if $\mathcal{A} \trianglelefteq_\pi \mathcal{B}$ then for all $x, y \in S_{\mathcal{B}}^{\mathbb{Z}}$ we have the inequality $d(\pi(x), \pi(y)) \leq d(x,y)$. Moreover, for any $y_1 \in S_{\mathcal{A}}^{\mathbb{Z}}$ there is some $y_2 \in S_{\mathcal{B}}^{\mathbb{Z}}$

such that $\pi(y_2) = y_1$ and $d(\pi(x), y_1) = d(x, y_2)$ (choose y_2 so that it equals x on the cells around position 0). Hence, if x is an equicontinuous point for $\mathcal{B}$ then $\pi(x)$ is an equicontinuous point for $\mathcal{A}$.

Finally, for the third assertion, it is sufficient to notice that the property of expansivity is defined by a formula using only universal quantifications on configurations so it remains true on a subset of configurations. ■

$\mathcal{T}_2$ is not an ideal for $\preccurlyeq_i$ and neither for $\preccurlyeq_m$ as shown by the following example.

Example 4.1. *Consider $\mathcal{B} \in \mathcal{T}_3$ of radius 1 and let $\mathcal{A}$ be the CA with radius 1, states set $S_\mathcal{A} = S_\mathcal{B} \cup \{M\}$ (with $M \notin S_\mathcal{B}$) with local rule $f_\mathcal{A}$ defined by*

$$f_\mathcal{A}(x, y, z) = \begin{cases} f_\mathcal{B}(x, y, z) & \text{if } \{x, y, z\} \subseteq S_\mathcal{B}, \\ y & \text{else.} \end{cases}$$

$\mathcal{B} \sqsubseteq \mathcal{A}$ so $\mathcal{B} \preccurlyeq_i \mathcal{A}$. However $\mathcal{A} \in \mathcal{T}_2$ since the configuration ${}^\omega M^\omega$ is an equicontinuous point. ■

Notice alos that $\mathcal{T}_3$ cannot be an ideal because $\sigma_1 \times \sigma_{-1} \in \mathcal{T}_3$ simulates $\sigma \in \mathcal{T}_1$.

Open Problem 2. *Are there $\mathcal{A} \notin \mathcal{T}_4$ and $\mathcal{B} \in \mathcal{T}_4$ such that $\mathcal{A} \preccurlyeq_s \mathcal{B}$ (i.e. the simulator CA is expansive up to rescaling but the simulated CA is not expansive, even up to rescaling)?*

4.3. (Un)decidability

The fact that many properties related to the simulation quasi-orders are undecidable comes with no surprise. For instance the nilpotency property, which is an undecidable problem [17], corresponds to an equivalence class in the 3 quasi-orders (theorem 3.1). However, there are non-trivial properties of these quasi-orders which are decidable (see below) and the edge between decidable and undecidable properties is hard to catch.

In this section, we consider two kind of problems in simulation quasi-orders: lower bounds (being above some fixed CA or set of CA) and upper bounds (being simulated by some fixed CA or some CA from a fixed set).

Theorem 4.9 ([22]). *The set of CA of radius 1 with a spreading state and nilpotent over periodic configurations is not co-recursively enumerable.*

Theorem 4.10. *Let $\mathcal{A}$ be any CA which is not nilpotent over periodic configurations. Let $\preccurlyeq$ be either $\preccurlyeq_i$ or $\preccurlyeq_m$. Then the set of CA $\mathcal{B}$ such that $\mathcal{A} \preccurlyeq \mathcal{B}$ is not co-recursively enumerable.*

Proof. We describe a computable transformation which, given a CA $\mathcal{C}$ of radius 1 with a spreading state, produce a CA $\mathcal{B}$ with the following properties:

- if $\mathcal{C}$ is not nilpotent over periodic configurations then $\mathcal{A} \preccurlyeq \mathcal{B}$;
- if $\mathcal{C}$ is nilpotent over periodic configurations then so is $\mathcal{B}$.

The theorem follows by theorem 4.3 since we have reduced the problem '$\mathcal{A} \preccurlyeq \mathcal{B}$?' to the problem of nilpotency over periodic configurations (reduced to CA of radius 1 with a spreading state).

We now describe the construction of $\mathcal{B}$ from $\mathcal{C}$. Suppose $\mathcal{C}$ has a spreading state q. $\mathcal{B}$ is the CA of radius 1 and states set $S_\mathcal{B} = (S_\mathcal{C} \setminus \{q\}) \times S_\mathcal{A} \cup \{q\}$ with local rule $f_\mathcal{B}$ defined by:

$$f_\mathcal{B}(a,b,c) = \begin{cases} \big(f_\mathcal{C}(a_1,b_1,c_1), f_\mathcal{A}(a_2,b_2,c_2)\big) & \text{if } \begin{cases} a,b,c \in S_\mathcal{B} \setminus \{q\} \text{ and} \\ f_\mathcal{C}(a_1,b_1,c_1) \neq q, \end{cases} \\ q & \text{in any other case,} \end{cases}$$

where a_i, b_i and c_i represent component i of a, b and c. Any periodic configuration c of $\mathcal{B}$ either leads to the uniform configuration $\overline{q}$, or contain a periodic configuration of $\mathcal{C}$ in its first component. Hence, if $\mathcal{C}$ is nilpotent over periodic configurations, then so is $\mathcal{B}$ (because q is precisely the spreading state of $\mathcal{C}$). If $\mathcal{C}$ is not nilpotent over periodic configurations, then there is a word $u \in (S_\mathcal{C} \setminus \{q\})^m$ and an integer $t \geq 1$ such that the periodic configuration c of period u verifies $G_\mathcal{C}^t(c) = c$. Therefore, by definition of $\mathcal{B}$, we have $\mathcal{A}^{\langle m,1,t,0\rangle} \sqsubseteq_i \mathcal{C}^{\langle m,1,t,0\rangle}$ where $i : S_\mathcal{A}^m \to S_\mathcal{B}$ is defined by:

$$i(a_1,a_2,\dots,a_m) = \big((a_1,u_1),\dots,(a_m,u_m)\big).$$

■

This result shows that it is generally undecidable to know whether a CA is lower-bounded by a given (fixed) one. However, there are noticeable exception in dimension 1 for $\preccurlyeq_s$ and $\preccurlyeq_m$.

Theorem 4.11. *Let $\preccurlyeq$ be either $\preccurlyeq_s$ or $\preccurlyeq_m$ and let $\mathcal{A}$ be a nilpotent CA. Then the problem of determining if a given $\mathcal{B}$ is above $\mathcal{A}$ for $\preccurlyeq$ is decidable.*

Proof. We are going to show that $\mathcal{A} \preccurlyeq \mathcal{B}$ if and only if $\mathcal{B}$ is not surjective and the theorem follows by decidability of surjectivity in one dimension [1]. First, by theorem 4.3, if $\mathcal{B}$ is surjective then $\mathcal{A} \not\preccurlyeq \mathcal{B}$. Suppose now that $\mathcal{B}$ is not surjective, *i.e.* that $\mathcal{B}$ possesses some Eden word $u \in S_\mathcal{B}^m$ for some length m. Then, denoting by $\mathcal{C}$ the CA over states set $S_\mathcal{C} = \{0,1\}$ which is constant equal to 0, we have $\mathcal{C} \trianglelefteq_\pi \mathcal{B}^{\langle m,1,1,0\rangle}$ if $\pi : S_\mathcal{B}^m \to S_\mathcal{A}$ verifies $\pi(w) = 0$ if and only if $w = u$. We deduce by theorem 3.1 that $\mathcal{A} \preccurlyeq \mathcal{B}$. ■

Open Problem 3. *Is there a non-surjective CA $\mathcal{A}$ which cannot injectively simulate any nilpotent CA? Is the problem of being above the class of nilpotent CA for injective simulation a decidable problem?*

Concerning upper-bound problems, the edge between decidability and undecidability is also non-trivial. For instance, theorem 4.5 shows the existence of a CA $\mathcal{A}$ such that the upper-bound decision problem '$\mathcal{B} \preccurlyeq \mathcal{A}$?' is decidable in dimension 1.

5. Tops of the Orders

In this section we study the maximal elements of the quasi-orders. This CA are able to simulate any other CA

Definition 5.1. *Let $\preccurlyeq$ be any relation among $\preccurlyeq_i$, $\preccurlyeq_s$ and $\preccurlyeq_m$. A CA $\mathcal{A}$ is said* $\preccurlyeq$-universal *if for any $\mathcal{B}$ we have $\mathcal{B} \preccurlyeq \mathcal{A}$. It is* strongly $\preccurlyeq$-universal *if it strongly $\preccurlyeq$-simulates any other CA.*

The notion of strong $\preccurlyeq_i$-universality above is exactly the same notion as *intrinsic universality* defined in section 5 of [7] and has already been considered several times in the literature (see [29] for a survey). In fact, strong and general universality are the same notion for $\preccurlyeq_i$ and $\preccurlyeq_m$.

Theorem 5.1. *There exist strongly $\preccurlyeq_i$-universal CA and all $\preccurlyeq_i$-universal CA are strongly $\preccurlyeq_i$-universal. The same is true for $\preccurlyeq_m$.*

Proof. For the existence of strongly $\preccurlyeq_i$-universal CA, see [29]. The theorem follows by application of theorem 12 of [7]. ■

Of course, any $\preccurlyeq_i$-universal is also $\preccurlyeq_m$-universal. The converse is an open problem.

Open Problem 4. *Do the notions of $\preccurlyeq_i$-universality and $\preccurlyeq_m$-universality coincide?*

Concerning $\preccurlyeq_s$, the situation is different: no CA is strongly $\preccurlyeq_s$-universal[5].

Theorem 5.2. *There is no strongly $\preccurlyeq_s$-universal CA.*

Proof. Suppose by contradiction that there is some strongly $\preccurlyeq_s$-universal $\mathcal{A}$. Consider a uniform configuration c of $\mathcal{A}$. There is n such that the orbit of c under $\mathcal{A}$ contains n different configurations (the orbit is ultimately periodic). Now consider $\mathcal{B}$ with $n+1$ states such that its uniform configurations are all in the same cycle of length $n+1$. By hypothesis, for any $\mathcal{B}$ there is some geometric transform α such that $\mathcal{B} \trianglelefteq_s \mathcal{A}^{\langle\alpha\rangle}$. Let d be the corresponding configuration of c for $\mathcal{A}^{\langle\alpha\rangle}$. The orbit of d contains at most n different configurations and it is therefore the same for the orbit of $s(d)$ under $\mathcal{B}$. But $s(d)$ is necessarily uniform and we get a contradiction with the choice of $\mathcal{B}$. ■

The theorem 12 of [7] don't apply for $\preccurlyeq_s$. However, we are not able either to construct a $\preccurlyeq_s$-universal CA, or to prove that there is no.

Open Problem 5. *Is there a $\preccurlyeq_s$-universal CA?*

For the rest of this section, we consider only $\preccurlyeq_i$ and $\preccurlyeq_m$.

5.1. On the Ways to Reach the Top

Universal CA are not hard to construct and the property of being universal is recursively enumerable since simulation relations considered here are recursively enumerable. However universality is not co-recursively enumerable as shown by the following theorem. The case of $\preccurlyeq_i$-universality was proven in [28]. Using theorem 4.10, the proof below is direct and includes the case of $\preccurlyeq_m$.

[5]The proof of this fact was suggested by G. Richard.

Theorem 5.3. *The set of $\preccurlyeq_i$-universal CA is not co-recursively enumerable and neither is the set of $\preccurlyeq_m$-universal CA.*

PROOF. There exists a CA which is $\preccurlyeq_i$-universal but not nilpotent over periodic configuration. To see this consider any universal CA and add a new state which is spreading: the resulting CA, say $\mathcal{A}$, contains at least two disjoint periodic orbits of periodic configurations and is thus not nilpotent over periodic configurations. The theorem follows by application of theorem 4.10 to $\mathcal{A}$ since $\mathcal{A}$ is by construction both $\preccurlyeq_i$-universal and $\preccurlyeq_m$-universal. ■

This result has some consequences on the structure of simulation quasi-orders 'near' the top. The following theorem indeed shows that a non-universal CA is always 'infinitely far' from the class of universal ones.

Theorem 5.4. *Let $\preccurlyeq$ be $\preccurlyeq_i$ or $\preccurlyeq_m$. And let $\mathfrak{U}$ be the set of $\preccurlyeq$-universal CA. Then we have:*

1. $\mathcal{A} \times \mathcal{B} \in \mathfrak{U} \iff \mathcal{A} \in \mathfrak{U}$ *or* $\mathcal{B} \in \mathfrak{U}$,
2. *if* $\mathcal{A} \notin \mathfrak{U}$ *then there is* $\mathcal{B} \notin \mathfrak{U}$ *with* $\mathcal{A} \preccurlyeq \mathcal{B}$ *but* $\mathcal{B} \not\preccurlyeq \mathcal{A}$.

PROOF.

1. By theorem 2.3 we have $\mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{B}$ and $\mathcal{B} \preccurlyeq \mathcal{A} \times \mathcal{B}$ which proves one direction. Moreover, there exists $\mathcal{C} \in \mathfrak{U}$ with 2 states only [2, 27]. If we suppose $\mathcal{A} \times \mathcal{B} \in \mathfrak{U}$ then, by theorem 5.1, it strongly simulates $\mathcal{C}$. Hence, by theorem 4.1, we have either $\mathcal{C} \preccurlyeq \mathcal{A}$ or $\mathcal{C} \preccurlyeq \mathcal{B}$ and thus either $\mathcal{A} \in \mathfrak{U}$ or $\mathcal{B} \in \mathfrak{U}$.
2. let $\mathcal{A} \notin \mathfrak{U}$. If $\mathcal{A}$ was such that $\mathcal{C} \preccurlyeq \mathcal{A}$ for all $\mathcal{C} \notin \mathfrak{U}$ then we would have $\mathfrak{U} = \{\mathcal{C} : \mathcal{C} \preccurlyeq \mathcal{A}\}$ and $\mathfrak{U}$ would be co-recursively enumerable contradicting theorem 5.3. So there is $\mathcal{C} \notin \mathfrak{U}$ with $\mathcal{C} \not\preccurlyeq \mathcal{A}$. To conclude the proof it is sufficient to choose $\mathcal{B} = \mathcal{A} \times \mathcal{C}$ (theorem 2.3). ■

5.2. Necessary But Not Sufficient Conditions

The purpose of this section is twofold. It compares the notions of universality defined above to other definitions of the literature and, by doing this, presents tools and techniques to prove non-universality of some CA (other proofs of non-universality for other purposes are developped in section 6).

One of the techniques we use to ensure that some CA is not universal yet achieving some behaviour B, is to add a spreading state and let the CA generate this state if it detects somewhere that the current configuration doesn't correspond to a 'legal' configuration, *i.e.* a configuration occuring normally when producing the behaviour B. Proofs of non-universality with this technique rely on the lemma below. Before stating and proving the lemma, we need to give some precision on spreading states and set of configurations 'supporting' a simulation.

First, the notion of spreading state is sensitive to the choice of the syntactical representation of the CA because it depends on the choice of the neihbourhood. In the sequel we say a CA $\mathcal{A}$ has a spreading state κ if any cell changes to state κ when κ appears in its minimal neighbourhood (*i.e.* the minimal set of cells upon which the local rule effectively depends).

Second, given a relation of the form $\mathcal{A} \sqsubseteq_i \mathcal{B}^{\langle m,1,t,z\rangle}$, there is an isomorphism between $(\mathcal{A}, S_{\mathcal{A}}^{\mathbb{Z}})$ and $\left(\mathcal{B}^{\langle m,1,t,z\rangle}, (i(S_{\mathcal{A}}))^{\mathbb{Z}}\right)$ as dynamical systems. At the level

of $\mathcal{B}$, the configurations involved in this relation is the set X of configurations made of infinite concatenation of elements of $i(S_\mathcal{A}) \subseteq S_\mathcal{B}^m$ (viewed as words of length m over alphabet $S_\mathcal{B}$). This kind of sets are called *bloc subshifts* and discussed in more details in section 3.2 of [7]. In the sequel, such a set X is called the *support* of the simulation.

Lemma 5.1. *Let $\mathcal{A}$ be a CA without spreading state and $\mathcal{B}$ be a CA with a spreading state κ. If $\mathcal{B}$ strongly $\preccurlyeq_m$-simulates $\mathcal{A}$, then the support X of the simulation cannot contain κ.*

Proof. By hypothesis, there are parameters m, t, τ, z and a CA $\mathcal{C}$ such that

$$\mathcal{A} \trianglelefteq_\pi \mathcal{C} \sqsubseteq_i \mathcal{B}^{\langle m,\tau,t,z\rangle}.$$

By choice of $\mathcal{B}$, $\mathcal{B}^{\langle m,\tau,t\rangle}$ admits κ^m as spreading state. Moreover, by definition of $\trianglelefteq$, the minimal neighbourhood of $\mathcal{A}$ is included in the minimal neighbourhood of $\mathcal{B}^{\langle m,\tau,t\rangle}$. Thus, if κ appears in some configuration of X then the state $\pi(i^{-1}(\kappa^m))$ is a spreading state for $\mathcal{A}$ because κ^n also appears in X for arbitrarily large n. ■

We first study how embeddings of Turing machines into CA can relate the notions of universality for Turing machines to the notions of universality derived from quasi-orders as defined above.

An embedding of a Turing machine $\mathcal{M}$ into a CA $\mathcal{A}$ is an embedding of the instantaneous descriptions of $\mathcal{M}$ into configurations of $\mathcal{A}$ such that instantaneous descriptions of successive steps of $\mathcal{M}$ corresponds to successive steps of $\mathcal{A}$ via the embedding. We don't give any formal definition of embedding since we will never prove negative result (*i.e.* assertions of the form 'there is no embbeding of $\mathcal{M}$ such that...'). However, the embeddings we use in the sequel are classical and already appeared in the literature (see [32]).

Theorem 5.5. *For any Turing machine $\mathcal{M}$, there exists a CA $\mathcal{A}$ which embeds $\mathcal{M}$ but is not $\preccurlyeq_m$-universal.*

Proof. Let $\mathcal{M} = (S_\mathcal{M}, Q_\mathcal{M}, \phi_\mathcal{M})$ where $S_\mathcal{M}$ is the set of states of $\mathcal{M}$, $Q_\mathcal{M}$ is the tape alphabet, and

$$\phi_\mathcal{M} : S_\mathcal{M} \times Q_\mathcal{M} \to S_\mathcal{M} \times Q_\mathcal{M} \times \{-1, 0, 1\}$$

is the transition function of $\mathcal{M}$. We construct a CA $\mathcal{A}$ over state set

$$S_\mathcal{A} = Q_\mathcal{M} \times \{\leftarrow, \rightarrow\} \cup Q_\mathcal{M} \times S_\mathcal{M} \cup \{\kappa\}$$

where $\rightarrow$ and $\leftarrow$ are states not already in $S_\mathcal{M}$. Each cell of $\mathcal{A}$ corresponds to a tape position of $\mathcal{M}$: it contains a letter from the tape alphabet and either a head with its current state or no head but an indication $\leftarrow$ or $\rightarrow$ telling in which direction to find the head. On configurations containing a single head, $\mathcal{A}$ mimics transitions of $\mathcal{M}$ step by step as expected. Thus, $\mathcal{A}$ embbeds $\mathcal{M}$. In addition, $\mathcal{A}$ checks that $\leftarrow$ a never occur to the left of a state from $S_\mathcal{M}$ or a $\rightarrow$ (and symmetrically for $\rightarrow$). If the check fails, then the state κ is generated and spreads.

This construction ensures that, for any initial configuration c, if the orbit of c never contains an occurrence of κ then it contains at most one head. Hence,

these orbits are such that at any time step state changes occur on the neighbourhood of at most one position (a head move involves a state change in two adjacent cells).

Now suppose that $\mathcal{A}$ is $\preccurlyeq_m$-universal and consider the CA $\mathcal{B} = \sigma_1 \times \sigma_{-1}$. $\mathcal{A}$ strongly simulates $\mathcal{B}$ (theorem 5.1). Since $\mathcal{B}$ has no spreading state, then the set X of configuration of $\mathcal{A}$ on which the simulation occurs never contains κ. We deduce that all orbits of configuration from X have the property described above. This implied that $\mathcal{B}$ is such that on all its orbits, at most 2 cells change their state between two steps: this in contradiction with the choice of $\mathcal{B}$. ■

Turing-universality of cellular automata is a fairly vague notion in the literature. We don't give a formal definition here since we won't prove any negative result concerning Turing-universality. We just consider that a CA able to embed a universal Turing machine[6] is Turing-universal.

We can choose $\mathcal{M}$ to be universal in the previous theorem (theorem 5.5). In this case, since the embedding used in the proof ensures that $\mathcal{M}$ is simulated in real time by $\mathcal{A}$, we deduce that the following problem is P-complete:

Input: a state $q \in S_{\mathcal{A}}$, an integer $t \geq 1$, and a word $u \in S_{\mathcal{A}}^{2rt+1}$ where r is the radius of $\mathcal{A}$;

Query: do we have $\mathcal{A}^t(u) = q$?

This problem of finite triangle computation has been considered several times in the literature and it has been proven that it was P-complete for particular CA [12, 26]. This notion of complexity inherited from sequential computation theory fails to capture the notion of universality associated to simulation quasi-orders.

Corollary 5.1. *There exists a CA which is Turing-universal and P-complete but not $\preccurlyeq_m$-universal.*

6. Induced Orders

This sections aims at studying particular CA or sets of CA for the ordered structure they induce in the simulation quasi-orders. While studying various properties of the quasi-orders in the previous sections, we have already established the existence or several induced infinite structures.

For instance, theorem 5.4 allows to construct an infinite strictly increasing chain of non-universal CA starting from any non-universal CA for the quasi-orders associated to $\preccurlyeq_i$ and $\preccurlyeq_m$. Besides, theorem 3.4 implies the existence of inifinite chains in the 3 quasi-orders studied in this paper.

Section 6.1 below gives a way to construct chains of length $\omega + \omega$ and an hint about the existence of chains of length $\omega \times \omega$. However, we leave open the question of the longest chain induced in any of the quasi-orders. We don't even know if one of them admits a dense chain.

Open Problem 6. *Does one of the quasi-orders admit a dense induced order?*

[6] We don't give any formal notion of universality for Turing machine either. In fact, we only need to suppose the existence of at least one universal Turing machine.

6.1. Limit Cartesian Product

We have seen in theorem 5.4 that if $\mathcal{A}$ is not universal, then $\mathcal{A} \times \mathcal{A}$ cannot be universal. Therefore, no finite Cartesian product of $\mathcal{A}$ with itself can be universal. Therefore, we have a chain of non-universal CA:

$$\mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{A} \times \mathcal{A} \preccurlyeq \cdots$$

For some $\mathcal{A}$, the chain collapses in a single equivalence class, *e.g.* if $\mathcal{A}$ is a translation (see lemma 3.4). However, the following theorem shows that for some $\mathcal{A}$, the chain is strictly increasing. Moreover, $\mathcal{A}$ can be chosen so that it embeds any Turing machine.

Theorem 6.1. *For any Turing machine $\mathcal{M}$, there is a CA $\mathcal{A}$ which embeds $\mathcal{M}$ and such that for any $1 \leq n < m$, one has:*

$$\underbrace{\mathcal{A} \times \cdots \times \mathcal{A}}_{m} \not\preccurlyeq_m \underbrace{\mathcal{A} \times \cdots \times \mathcal{A}}_{n}.$$

Proof. Le $\mathcal{A}$ be the CA constructed in the proof of theorem 5.5. We can suppose that $\mathcal{M}$ is such that it can produce infinite sequences of left move of its head starting from a blank tape and leaving the tape blank, and the same for right moves (if $\mathcal{M}$ has not this property, just add some states to achieve this behaviour).

Denote by $\mathcal{B}_m$ the product of m copies of $\mathcal{A}$ and by $\mathcal{B}_n$ the product of n copies. We can construct for any set of positions $z_1, \ldots, z_m$ a configuration c of $\mathcal{B}_m$ such that for all i the ith component contains a correct instantaneous descritpion of $\mathcal{M}$ where the head is at position z_i in a state suitable to generate an infinite sequence of left or right moves. Now let c' be a onfiguration of $\mathcal{B}_n$ corresponding to c via simulation. First, if some component i of c' contains a spreading state, it will spread and, after some time t, will be present at some position where the configuration $G^t_{\mathcal{B}_m}(c)$ contains no head, but only a blank tape symbol on each component. This means that blocs of blank tape symbols in $\mathcal{B}_m$ can be simulated by blocs of $\mathcal{B}_n$ where the ith component is a bloc of spreading states. Considering again the orbit of c, we deduce that it can be simulated by a configuration c'' where the ith component is everywhere a spreading state except at a finite number of positions. Thus after some time, the ith component will become uniform and constant. It is then straightforward to show that it is useless for the simulation and that in fact $\mathcal{B}_m$ on c can be simulated by only $n-1$ copies of $\mathcal{A}$.

Applying the reasoning inductively, we can therefore suppose that no spreading state appears on any component in the orbit of the configuration c' defined above. Since, the orbit of c is such that there are m distant position where some state change at each step, it must be the case in the orbit of c'. Since, $n < m$, there must be some component with two heads and therefore a spreading state must appear after the first step: this is in contradiction with what we have just supposed. ■

For the CA $\mathcal{A}$ of the previous theorem, we can ask if the infinite chain of Cartesian product is upper-bounded by some non-universal CA, or if any CA able to simulate each product of the chain is necessarily universal. One can

imagine that for a sufficiently simple $\mathcal{A}$, there is some room above the chain of products of $\mathcal{A}$ and below the class of universal CA.

The rest of this section is devoted to the proof of a stronger result: for any $\mathcal{A}$, there is a CA $\mathcal{B}$ which is able to simulate any finite product of $\mathcal{A}$ and such that $\mathcal{B}$ is universal if and only if $\mathcal{A}$ is universal. Moreover, $\mathcal{B}$ can be obtained from $\mathcal{A}$ constructively. Because it extends property of Cartesian product given by theorem 5.4, this construction will be called *limit product* in the sequel. If $\mathcal{A}$ is a CA, its limit product is denoted by $\mathcal{A}_\infty$.

Note: In the rest of this section we only consider the simulation $\preccurlyeq_m$.

Without loss of generality, we can suppose that $\mathcal{A}$ has radius 1 (theorem 2.3). To be able to simulate the product $\mathcal{B}$ of n copies of $\mathcal{A}$, $\mathcal{A}_\infty$ is made of 3 layers (its state set is a Cartesian product union a single state, which is a spreading state as explain hereafter):

1. the *state* layer,
2. the *transport* layer, and
3. the *synchronisation* layer.

and proceeds as follows.

State layer. Each components of a cell of $\mathcal{B}$ is simulated by a bloc of 3 adjacent cells in the state layer of $\mathcal{A}_\infty$. More precisely, component i ($0 \leq i \leq n-1$) of cell z of $\mathcal{B}$ is simulated by the bloc of 3 cells of $\mathcal{A}_\infty$ beginning at position $3(nz+i)$. This bloc is referred to as $\mathtt{B}_{z,i}$ in the sequel. In $\mathtt{B}_{z,i}$, the center cell stores the ith component of the cell z of $\mathcal{B}$ and the two other are used to store temporarilly ith components of cell $z-1$ and $z+1$.

Transport layer. The role of the transport layer is precisely to bring states of ith components corresponding to cell $z-1$ and $z+1$ of $\mathcal{A}$ to the dedicated cells of $\mathcal{A}_\infty$ in $\mathtt{B}_{z,i}$. Then, the transition $f_{\mathcal{A}}(x_{z-1}, x_z, x_{z+1})$ of the ith component of $\mathcal{B}$ can be simulated locally by $\mathcal{A}_\infty$ in $\mathtt{B}_{z,i}$. Transport is done in parallel for any i and any z. To do this, the transport layer is made of a succession of particles (one every 3 cells), each one being able to carry a state of $\mathcal{A}$. Initially aligned with the center of blocs, the particules move in parallel according to a cycle of 5 steps:

1. move right by $3n$ cells and read the state seen on the state layer;
2. move left by $3n-1$ cells and write the memorized state on the state layer;
3. move left by $3n+1$ cells and read the state seen on the state layer;
4. move left by $3n-1$ cells and write the memorized state on the state layer;
5. move 1 cell right and apply local rule $f_{\mathcal{A}}$ on state layer at the current position;

Synchronisation layer. The role of the synchronisation layer is to orchestrate the cycle of particle moves and it must be able to do it for arbitrary large values of n (simulating arbitrarily large Cartesian products of $\mathcal{A}$ is sufficient to simulate all products of $\mathcal{A}$). It contains a flag that can take one of the 4 indications 'left', 'right', 'read', 'write' and 'transition'. The flag is changed everywhere synchronously according to a cycle suitable to ensure that particles of the transport layer produce the cycle described above when they follow the instruction given by the flag.

We now describe in detail the synchronisation layer. Denote by u_n the flag sequence mentionned above in the simulation of a product of n copies of $\mathcal{A}$, and let E be the set of flag states.

Theorem 6.2. *There is a CA $\mathcal{C}$ with a spreading state κ and a map $\pi : S_{\mathcal{C}} \to E$ such that $\mathcal{C}$ is not $\preccurlyeq_m$-universal, and, for any configuration $c \in S_{\mathcal{C}}^{\mathbb{Z}}$, one of the following property is true:*

Cycle: *at each time in the orbit of c, all cells have the same image by π and the sequence with time of this common image is periodic of period u_n for some n;*

Frozen: *at each time in the orbit of c, all cells have the same image by π, but this common image remains constant after a certain time;*

Error: *the spreading states appears at some time in the orbit of c.*

Moreover $\mathcal{C}$ is such that there are configurations having the 'cycle' property above producing period u_n for arbitrarily large n.

Proof. First, notice that flag changes in the sequence u_n are separated by a number of steps which is either constant (independant of n), or of the form $3n + c$ with c a constant (we can suppose $c \geq 0$ without loss of generality). To simplify notations, we will suppose in this proof that u_n alternates between two values 0 and 1 every $3n$ steps. Adapting the proof for the real u_n is just a matter of adding a finite set of special states to deal with constants.

The proof is based on a reversible solution $\mathcal{B}$ to the firing squad synchronization problem proposed by K. Imai and K. Morita: in [16], they construct a reversible CA $\mathcal{B}$ with a subset of states F (the firing states) such that for any n, there is a periodic configuration c_n verifying[7]:

- $G_{\mathcal{B}}^{3n}(c_n) \in F^{\mathbb{Z}}$
- $G_{\mathcal{B}}^{t}(c_n) \in (S_{\mathcal{B}} \setminus F)^{\mathbb{Z}}$ for all t, $0 \leq t < 3n$.

Without loss of generality, we can suppose that $\mathcal{B}$ and its inverse are syntactically represented with the same radius. We now define a CA $\mathcal{C}_0$ of radius r, with states set $S_{\mathcal{C}_0} = S_{\mathcal{B}} \times S_{\mathcal{B}} \times \{0, 1\}$, and with transition function:

$$f_{\mathcal{C}_0}\big((a_{-r},a'_{-r},b_{-r}),\dots,(a_r,a'_r,b_r)\big) = \begin{cases} \big(f_{\mathcal{B}}(a_{-r},\dots,a_r), f_{\mathcal{B}^{-1}}(a'_{-r'},\dots,a'_{r'}), \chi(a_0,b_0)\big) & \text{if } b_0 = 1, \\ \big(f_{\mathcal{B}^{-1}}(a_{-r},\dots,a_r), f_{\mathcal{B}}(a'_{-r},\dots,a'_r), \chi(a'_0,b_0)\big) & \text{if } b_0 = 0, \end{cases}$$

where $\chi(a, b)$ equals $1 - b$ if $a \in F$ and b else. Intuitively, on configurations where the third component is uniform equal to b, $\mathcal{C}_0$ mimic $\mathcal{B}$ on the first component and $\mathcal{B}^{-1}$ on the second one if $b = 1$ or the converse if $b = 0$. Moreover,

[7] In [16], the main concern is synchronisation of finite segments of cells surrounded by a quiescent state. To extend the property to infinite configurations, it is crucial that "garbage" (which must be conserved to ensure reversibility) do no spread outside the initial segment. The solution of K. Imai and K. Morita has precisely this property as it is explicitely mentionned in [16].

the value of b is switched each time the component playing $\mathcal{B}$ encounters a firing state. Hence, if we choose for π the projection on third component, $\mathcal{C}_0$ started from configurations c_n has the property 'cycle' and produce the periodic sequence u_n.

We now enrich $\mathcal{C}_0$ with a spreading state which is produced each time one of the following local checking fails:

- the third component $\{0,1\}$ must be uniform;
- for the two first components, a state from F (firing state) must always be surrounded by states from F only;
- states from F are forbidden on the second component if $b = 1$ and states from F in the first component are forbidden if $b = 0$.

The third condition ensures that in the case of a 'cycle' regime (firing states appearing infinitely often), the period is equally divided between steps where $b = 0$ and steps where $b = 1$. To ensure that such 'cycle' regime always produce an alternance of exactly $3n$ zeros and $3n$ ones, we add a component implementing a couter modulo 3: the value of this component is incremented modulo 3 at each step (whatever the context) and a spreading state is generated if a cell contains a firing state and the counter is not 0 modulo 3. Denote by $\mathcal{C}$ the CA obtained and consider any configuration c. If no spreading state appears in the orbit of c, then the third component is uniform. If it changes of state only a finite number of times, then we are in the 'frozen' regime. If there are infinitely many changes, it follows from the discussion above that the conditions of the 'cyclic' regime are fulfilled.

To conclude the theorem, it remains to prove that $\mathcal{C}$ is not $\preccurlyeq_m$-universal. Suppose by contradiction that it is and let $\mathcal{U}$ be any universal CA without spreading state and consider the set X of configurations of $\mathcal{C}$ which is the support of the strong simulation of $\mathcal{U}$ ($\mathcal{C}$ strongly simulmates $\mathcal{U}$ by theorem 5.1). X cannot contain any occurrence of the spreading state (by lemma 5.1), it implies that all configurations of X have a uniform third component. But, on such configuration, the dynamics of $\mathcal{C}$ is reversible. Hence $\mathcal{U}$ is reversible: this is a contradiction with its universality by theorem 4.3. ■

The synchronization layer of limit products are exactly the automaton $\mathcal{C}$ of the previous theorem, except that the spreading state of $\mathcal{C}$ now becomes a global spreading state. Before establishing the main result of this section, we give more details concerning the state layer and the transport layer of $\mathcal{A}_\infty$.

The state layer is made from state set $S_\mathcal{A} \times \{L, C, R\}$ where L, C and R are states to identify explicitely the role of each cell in each bloc $\mathtt{B}_{z,i}$: C for the cell storing the ith component of cell z of $\mathcal{A}$ and L and R to temporarilly store states of ith component of cells $z-1$ and $z+1$ respectively.

The transport layer is made from state set $S_\mathcal{A} \cup \{\perp\}$ where $\perp$ is the state used to separate particle carrying a state from $S_\mathcal{A}$.

So the states set of $\mathcal{A}_\infty$ is:

$$\underbrace{S_\mathcal{A} \times \{L, C, R\}}_{\text{state}} \times \underbrace{S_\mathcal{A} \cup \{\perp\}}_{\text{transport}} \times \underbrace{S_\mathcal{C} \setminus \{\kappa\}}_{\text{synchronization}} \cup \{\kappa\}.$$

In addition to the behaviour described above, $\mathcal{A}_\infty$ do the following local checkings and generates the spreading state κ if one of them fails:

- the second component of transport layer must be periodic of period LCR;
- the transport layer must contain an alternance of 1 state from $S_{\mathcal{A}}$ and 2 states $\perp$;
- when doing read and write operations, the particles of the transport layer must be aligned with the right type of state in the state layer:
  - type C when reading,
  - type R when writing at step 2,
  - type L when writing at step 4;
- when the synchronisation layer says 'transition', check that the particules are aligned with cell of type C in the state layer.

All those checking ensure the following property: if no spreading state is generated and if the component layer produce a correct cycle of instructions, then the behaviour of the state layer is equivalent to the behaviour of some Cartesian product of $\mathcal{A}$ (up-to some rescaling).

Before stating the main theorem, we establish a simple yet useful lemma saying that if $\mathcal{A}$ simulates $\mathcal{B}$ with support X, then everything $\mathcal{A}$ can simulate using a support included in X can also be simulated by $\mathcal{B}$.

Lemma 6.1. *Let $\preccurlyeq$ be either $\preccurlyeq_i$ or $\preccurlyeq_m$. Let $\mathcal{A}$ and $\mathcal{B}$ be such that the simulation $\mathcal{A} \preccurlyeq \mathcal{B}$ occur on a support X of configurations of $\mathcal{B}$. If $\mathcal{B}$ $\preccurlyeq$-simulate $\mathcal{C}$ on a support included in X, then $\mathcal{A}$ $\preccurlyeq$-simulates $\mathcal{C}$.*

Proof. We consider the case where $\preccurlyeq$ is $\preccurlyeq_i$. By hypothesis, we have $\mathcal{A}^{\langle\alpha\rangle} \sqsubseteq_i \mathcal{B}^{\langle\beta_1\rangle}$ on support X and $\mathcal{C}^{\langle\gamma\rangle} \sqsubseteq_j \mathcal{B}^{\langle\beta_2\rangle}$ on support $Y \subseteq X$. Now, let m_α, m_{β_1}, m_{β_2} and m_γ be the packing parameters of transforms α, β_1, β_2 and γ respectively. The injective maps i and j induce two injective maps i_{β_2} and j_{β_1} with the following domains and ranges:

$$i_{\beta_2} : S_{\mathcal{A}}^{m_\alpha m_{\beta_2}} \to S_{\mathcal{B}}^{m_{\beta_1} m_{\beta_2}}$$
$$j_{\beta_1} : S_{\mathcal{C}}^{m_\gamma m_{\beta_1}} \to S_{\mathcal{B}}^{m_{\beta_1} m_{\beta_2}}$$

Therefore $\phi = i_{\beta_2}^{-1} \circ j_{\beta_1}$ is a well-defined injective map from $S_{\mathcal{C}}^{m_\gamma m_{\beta_1}}$ into $S_{\mathcal{A}}^{m_\alpha m_{\beta_2}}$. Now define the transforms η_a and η_c to be the composition of α and β_2, and of γ and β_1 respectively. Then we have $\mathcal{C}^{\langle\eta_c\rangle} \sqsubseteq_\phi \mathcal{A}^{\langle\eta_a\rangle}$.

The extension of the previous reasoning to $\preccurlyeq_m$ is straightforward. ■

This lemma together with lemma 5.1 is the key to a kind of 'self-checking' simulation used in the construction of the limit product (and re-used in section 6.2). A 'self-checking' simulation of $\mathcal{B}$ by $\mathcal{A}$ is standard simulation of $\mathcal{B}$ by $\mathcal{A}$ on some support X with the additional property that $\mathcal{A}$ 'checks' locally on any configuration that it belongs to X, and triggers some pathological behaviour (typically a spreading state) in case of check failure. Hence any possible strong simulation of some $\mathcal{C}$ by $\mathcal{A}$ is such that:

- either it has a support included in X in which case $\mathcal{B}$ can also simulate $\mathcal{C}$ by lemma 6.1,

- or it must contain some $c \notin X$ in its support in which case a spreading state is genereated and lemma 5.1 give some limitation on $\mathcal{C}$.

To show that a spreading state is generated in the second case above, a crucial property is that the support of any simulation is by definition always irreducible: if u_1 and u_2 are words occuring in two configurations of the support, there exist a third configuration of the support where u_1 and u_2 both appear (see section 3.2 of [7] for a more detailed discussion on supports of simulations).

We now state the main theorem of this section.

Theorem 6.3. *For any $\mathcal{A}$, its limit product $\mathcal{A}_\infty$ is such that:*

- $\underbrace{\mathcal{A} \times \cdots \times \mathcal{A}}_{n} \preccurlyeq_m \mathcal{A}_\infty$ *for all* $n \geq 1$,
- $\mathcal{A}_\infty$ *is $\preccurlyeq_m$-universal if and only if $\mathcal{A}$ is $\preccurlyeq_m$-universal.*

Proof. The first assertion follows from the construction of $\mathcal{A}_\infty$ and the detailed discussion above. Now suppose that $\mathcal{A}_\infty$ is $\preccurlyeq_m$-universal and let $\mathcal{U}$ be any universal CA without spreading state. By theorem 5.1, $\mathcal{A}_\infty$ strongly simulates $\mathcal{U}$: $\mathcal{U} \trianglelefteq \mathcal{A}_\infty{}^{\langle\alpha\rangle}$ for some geometrical transform α. Let X denote the support of the simulation. By choice of $\mathcal{U}$, the spreading state κ cannot appear in any orbit of any configuration of X (by lemma 5.1). We deduce from theorem 6.2 that the synchronization component is in the same regime (either 'cycle' for a fixed value n or 'frozen') for all the configurations of X because otherwise, we could construct a configuration in X producing a spreading (by irreducibility of X).

In the case where all configurations are in the frozen regime, the flag of the synchronization layer becomes constant after some time t_0, so the transport layer has the behaviour of a translation (or identity) and the state layer remains constant. t_0 is identical for all configurations of X (because otherwise, we could once more combine two configurations to produce a spreading state, by irreducibility of X). Then, consider a CA $\mathcal{U}_+$ with state set $S_\mathcal{U} \times \{0, \dots, t_0\}$ which has the following behaviour:

- the second component is decreased by one until it reaches 0;
- on the first component, the local rule of $\mathcal{U}$ is applied, but only if the second component is 0.

Since $\mathcal{U}$ is $\preccurlyeq_m$-universal, it can strongly simulate $\mathcal{U}_+$ (by theorem 5.1): precisely, $\mathcal{U}_+ \trianglelefteq \mathcal{U}^{\langle m,1,t,z\rangle}$. Consider the set Y of configurations of $\mathcal{U}$ corresponding via simulation to the set of configurations of $\mathcal{U}_+$ uniformly equal to t_0 on the second component. Denote by $X_Y \subseteq X$ the corresponding set of configurations of $\mathcal{A}_\infty$. By choice of $\mathcal{U}_+$, we know that $\mathcal{U}$ simulates itself on the set of configurations $G_\mathcal{U}^{tt_0}(Y)$. This implies that for some $t' \geq t_0$, $\mathcal{A}_\infty$ can simulate U using as support the set of configuration $G_{\mathcal{A}_\infty}^{t'}(X_Y)$. By hypothesis, starting from such configurations, $\mathcal{A}_\infty$ has a behaviour of translation or identity on the state and transport layers. Since the synchronizing component evolves independantly of the others, we deduce by lemma 6.1 that there is some CA $\mathcal{B}$ which is a product of translation (corresponding to state and transport layers) such that $\mathcal{B} \times \mathcal{C}$ simulates $\mathcal{U}$: this is a contradiction by theorem 5.4 since neither $\mathcal{B}$ (theorem 3.4), nor $\mathcal{C}$ (theorem 6.2) is universal.

Hence, we are necessarily in the case where the synchronization layers produce a valid cycle. Since no spreading state can be generated in the orbit of any configuration of X, the state layer always behave like a Cartesian product of n copies of $\mathcal{A}$. The value of n is in fact common to all configurations of X (as shown above), so we deduce by lemma 6.1 that $\underbrace{\mathcal{A} \times \cdots \times \mathcal{A}}_{n}$ simulates $\mathcal{U}$ and $\mathcal{A}$ is therefore universal by theorem 5.4. ■

Of course, we can consider $\mathcal{A}_\infty$ itself as a new candidate for taking its finite Cartesian products and applying the limit product construction. In fact, the process can be repeated forever with the guarantee that no CA ever produced in this chain will be universal, provided the initial CA is not. However, there is no reason why this infinite chain should be strictly increasing. In particular, even if

$$\mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{A} \preccurlyeq \mathcal{A} \times \mathcal{A} \times \mathcal{A} \preccurlyeq \cdots$$

is a strictly increasing chain, it might be the case that $\mathcal{A}_\infty$ is equivalent to $\mathcal{A}_\infty \times \mathcal{A}_\infty$. Therefore we have only proven that one of the following properties is true:

- there is a strictly increasing chain of length $\omega \times \omega$ in the quasi-order $(AC, \preccurlyeq_m)$,
- for any non-universal CA $\mathcal{A}$, there is a non-universal CA $\mathcal{B}$ such that $\mathcal{A} \preccurlyeq_m \mathcal{B}$ and $\mathcal{B} \times \mathcal{B}$ is equivalent to $\mathcal{B}$.

6.2. Sub-Families of Cellular Automata

Theorem 2.3 shows that any equivalence class in any quasi-order contains some CA with radius 1. This fact is a direct consequence of a well-known transformation of CA with large radius into CA of smaller radius with more states (this transformation is called 'higher block presentation' in symbolic dynamics, see [20]).

It is also sometimes invoked in the literature that considering CA with 2 states only is not restrictive since there is a converse transformation that transforms a CA with many states into a CA with only 2 states but a larger radius. However, the situation is different since there are equivalence classes without CA having 2 states: *e.g.* $\mathcal{Z}_p$ for any prime $p \neq 2$ as shown by theorem 3.2 and lemma 3.2. Note that the same is true for any fixed state set of cardinal n: the equivalence class of $\mathcal{Z}_p$ contains no such CA provided p is prime and do not divide n.

Hence, this transformation introduces a bias: the transformed CA may be inequivalent to the original one. Meanwhile, we know that CA with 2 states can be as powerful as CA in general since there are universal CA with 2 states only [2, 27] (for simulation realtions $\preccurlyeq_i$ and $\preccurlyeq_m$). More precisely, as we will see below, the transformation applied to a universal CA always yields a universal CA because the transformed CA simulates the original one. Since the original and the transformed CA are not always in the same equivalence class, one question that naturally arises is: what CA can be simulated by the transformed CA but not by the original one? Although it provides only partial answers, this section is devoted to that kind of questions, for CA with 2 states and for other families.

Formally, given a familly $\mathfrak{F}$ of CA, we say a map $\phi : AC \to \mathfrak{F}$ is a $\preccurlyeq$-encoding of CA into familly $\mathfrak{F}$ if

$$\forall \mathcal{A}, \mathcal{A} \preccurlyeq \phi(\mathcal{A}).$$

We will only consider simulation relations $\preccurlyeq_i$ and $\preccurlyeq_m$ in the sequel, thus an encoding into $\mathfrak{F}$ implies that there are universal CA in $\mathfrak{F}$. A trivial example of such an encoding is given by $\mathfrak{F} = \{\mathcal{U}\}$ where $\mathcal{U}$ is a universal CA and ϕ is the map sending any CA to $\mathcal{U}$. We are interested in using this notion of encoding with families which are more 'representative' of the diversity of behaviours in the whole set of CA. To express this we introduce the following notion of *faithfulness*.

Given a $\preccurlyeq$-encoding $\phi : AC \to \mathfrak{F}$ and a set E of CA, we say that ϕ is *faithful* for E if:

$$\forall \mathcal{B} \in E : \mathcal{B} \preccurlyeq \mathcal{A} \iff \mathcal{B} \preccurlyeq \phi(\mathcal{A}).$$

An encoding is faithful for E if the original CA and its image by the encoding simulate exactly the same CA in E. So, to give some evidence that a family $\mathfrak{F}$ is 'representative' of CA in general, we can exhibit an encoding of CA into $\mathfrak{F}$ which is faithful for a set E of CA as large as possible. When E is the whole set of CA, the faithfulness implies that there is a CA of family $\mathfrak{F}$ in any equivalence class: this is the case for CA with radius 1.

The next theorem gives 4 encodings which are faithful for U, the set of $\preccurlyeq_m$-universal CA. The families corresponding to these encodings were already defined in this paper except one: captive CA.

Captive CA where introduced in [33] and are defined by a simple restriction on the transition rule. A CA $\mathcal{A}$, of states set $S_\mathcal{A}$, radius r and local rule $f_\mathcal{A}$ is captive if:

$$\forall a_{-r}, \ldots, a_r \in S_\mathcal{A} : f_\mathcal{A}(a_{-r}, \ldots, a_r) \in \{a_{-r}, \ldots, a_r\}.$$

In the following theorem, encodings are different but their faithfulness rely on the same idea of 'self-checking' simulation explained above which uses lemma 5.1 and lemma 6.1.

Theorem 6.4. *Let $\preccurlyeq$ be $\preccurlyeq_i$ or $\preccurlyeq_m$. For any family of CA below, there is a $\preccurlyeq$-encoding from CA into $\mathfrak{F}$ wich is faithful for the set U of $\preccurlyeq_m$-universal CA:*

- *CA with 2 states,*
- *CA in $\mathcal{T}_2$,*
- *CA in $\mathcal{T}_3$,*
- *captive CA.*

Proof. To describe the encoding for each family, we suppose $\mathcal{A}$ is a CA with states set $S_\mathcal{A} = \{a_1, \ldots, a_n\}$, with radius r and location rule $f_\mathcal{A}$.

2 **states CA.** Let m be an integer large enough and ψ be an injective map from $S_\mathcal{A}$ to $\{0,1\}^m$ such that no word $\psi(a)$ contains an occurrence of 11. Now define the injective map $i : S_\mathcal{A} \to \{0,1\}^{m+4}$ by $i(a) = 0110\psi(a)$. Let $r' = (r+1)(m+4)$. $\phi(\mathcal{A})$ is a CA of radius r' and states set $\{0,1\}$ defined as follows:

- on the set X of configurations made of infinite concatenation of words from $i(S_{\mathcal{A}})$, $\phi(\mathcal{A})$ is isomorphic to $\mathcal{A}$ so that $\mathcal{A} \sqsubseteq_i \phi(\mathcal{A})$;
- everywhere else, $\phi(\mathcal{A})$ generates a 1.

The map ϕ is thus an encoding of CA into 2-states CA. Now suppose that $\phi(\mathcal{A})$ is universal and let $\mathcal{U}$ be a universal CA with 2 states and no spreading state which is strongly simulated by $\phi(\mathcal{A})$ on support Y (theorem 5.1). If there is some $y \in Y$ with $y \notin X$ then

- either there are two occurrences of 0110 in y which are not correctly spaced,
- or there is a word $0110u0110$ occurring in y with $u \notin \psi(S_{\mathcal{A}})$.

In any case, the image of y will contain an occurrence of 111 (because the above error must be seen by at least 3 consecutive cells) and 1's will propagate like a spreading state which is impossible by lemma 5.1 because otherwise $\phi(\mathcal{A})^{\langle 3,1,1,0\rangle}$ could simulate $\mathcal{U}$ on a support where it possesses the spreading state 111. So $Y \subseteq X$ and lemma 6.1 shows that $\mathcal{A}$ simulates $\mathcal{U}$. Hence $\mathcal{A}$ is universal if and only if $\phi(\mathcal{A})$ is.

Captive CA. The encoding technique for captive CA is very similar and already appeared in a non-faithful form in [33]. Let u be the word $a_1 \cdots a_n$, let $\#$ be a state not in $S_{\mathcal{A}}$ and denote $Q = S_{\mathcal{A}} \cup \{\#\}$. We define the injective map $i : S_{\mathcal{A}} \to Q^{n+3}$ by $i(a) = \#u\#a$. We then define $\phi(\mathcal{A})$ in a way similar to the case above. Its radius is $r' = (r+1)(n+3)$, its states set is Q and its local rule is such that:

- on the set X of configurations made of infinite concatenation of words from $i(S_{\mathcal{A}})$, $\phi(\mathcal{A})$ is isomorphic to $\mathcal{A}$ so that $\mathcal{A} \sqsubseteq_i \phi(\mathcal{A})$;
- everywhere else, $\phi(\mathcal{A})$ take as new state the maximum of its neighbours for some fixed ordering of Q such that $\#$ is the maximum.

First, $\phi(\mathcal{A})$ is captive and ϕ defines an encoding of CA into captive CA. Second, notice that for any support of simulation Y of $\phi(\mathcal{A})$, if there is some $y \in Y$ with $y \notin X$ then, by irreducibility of Y, either there is $y' \in Y$ with $y' \notin X$ and y' contains a $\#$, or $\#$ never appears in Y. In the second case, $\phi(\mathcal{A})$ always applies a max as local rule and therefore possesses a spreading state when restricted to Y. In the first case, consider the configuration y' and $z \in \mathbb{Z}$ such that positions z and $z+1$ both see a $\#$ in their neighbourhood and a local pattern not in X (such a z must exist by choice of y' and definition of X). Then $\phi(\mathcal{A})(y')$ contains the pattern $\#\#$ which is spreading by definition of $\phi(\mathcal{A})$. In any case we can apply the usual reasoning with lemma 5.1 and lemma 6.1: any CA without spreading state strongly simulated by $\phi(\mathcal{A})$ is also simulated by $\mathcal{A}$. So the encoding ϕ is faithful for universal CA.

$\mathcal{T}_2$ and $\mathcal{T}_3$. For $\mathcal{T}_2$, the encoding is simple: $\phi(\mathcal{A})$ is just $\mathcal{A}$ with an additional state κ which is spreading. The resulting CA $\phi(\mathcal{A})$ is always in $\mathcal{T}_2$ since κ^{2r} is a blocking word (see [18]). Lemma 5.1 is then sufficient to prove that it is an encoding from AC to $\mathcal{T}_3$ which is faithful for universal CA.

For $\mathcal{T}_3$, the proof is even simpler: $\phi(\mathcal{A}) = \mathcal{A} \times \sigma_1 \times \sigma_{-1}$ is always in $\mathcal{T}_3$ since $\sigma_1 \times \sigma_{-1} \in \mathcal{T}_3$ and an equicontinuous point in a Cartesian product induce equicontinuous points for each component. Theorem 5.4 concludes for the faithfulness.

■

These encodings allow to transport some properties of general CA concerning the top of quasi-orders into order structures induced by each family[8].

Corollary 6.1. *Let $\preccurlyeq$ be $\preccurlyeq_i$ or $\preccurlyeq_m$ and let $\mathfrak{F}$ be a family of CA among: CA with 2 states, $\mathcal{T}_2$, $\mathcal{T}_3$, captive CA. Then we have the following properties:*

- *the set of $\preccurlyeq$-universal CA in $\mathfrak{F}$ is not co-r.e.*
- *for any non-universal $\mathcal{A} \in \mathfrak{F}$, there is a non universal $\mathcal{B} \in \mathfrak{F}$ with $\mathcal{A} \preccurlyeq \mathcal{B}$ but $\mathcal{B} \not\preccurlyeq \mathcal{A}$.*

Proof. The first property is a direct corollary of theorem 6.4 and 5.3 by definition of faithful encodings.

For the second property, consider the encoding ϕ established in theorem 6.4 and let $\mathcal{A} \in \mathfrak{F}$ be any non-universal CA. By theorem 5.4, there is some non-universal CA $\mathcal{B}$ such that $\mathcal{A} \preccurlyeq \mathcal{B}$ but $\mathcal{B} \not\preccurlyeq \mathcal{A}$. By faithfulness of ϕ, $\phi(\mathcal{B}) \in \mathfrak{F}$ is not universal and by the definition of encoding it simulates $\mathcal{A}$ without being simulated by $\mathcal{A}$. ■

The families considered above induce structures sharing some properties with the general quasi-orders 'near the top'. However, the complete characterisation of equivalence classes occupied by some CA of these families is more challenging.

Open Problem 7. *What are the quivalence classes of the simulation quasi-orders containing a 2-states CA? a captive CA? a CA from $\mathcal{T}_2$? a CA from $\mathcal{T}_3$?*

References

[1] S. Amoroso, Y. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tesselation Structures, Journal of Computer and System Sciences 6 (1972) 448–464.

[2] E. R. Banks, Universality in cellular automata, in: Eleventh Annual Symposium on Switching and Automata Theory, IEEE, Santa Monica, California, 1970.

[3] G. Cattaneo, E. Formenti, L. Margara, Topological definitions of deterministic chaos: applications to cellular automata dynamics, in: Cellular automata (Saissac, 1996), Kluwer Acad. Publ., Dordrecht, 1999, pp. 213–259.

[8] A stronger result concerning captive CA appears in [34]: $\preccurlyeq_i$-universality is undecidable even if we restrict to captive CA with a fixed (but sufficiently large) radius.

[4] G. Cattaneo, E. Formenti, L. Margara, J. Mazoyer, A new shift-invariant metric on S^Z inducing a non-trivial topology, in: I. Privara, P. Rusika (eds.), Mathematical Foundations of Computer Science (MFCS'97), vol. 1295 of Lecture Notes in Computer Science, Springer-Verlag, 1997.

[5] K. Čulik, II, J. Pachl, S. Yu, On the limit sets of cellular automata, SIAM Journal on Computing 18 (4) (1989) 831–842.

[6] B. A. Davey, H. A. Priestley, Introduction to Lattices and Order, Cambridge University Press, 2002.

[7] M. Delorme, J. Mazoyer, N. Ollinger, G. Theyssier, Bulking I: an Abstract Theory of Bulking, *unpublished* (2008).

[8] J. Durand-Lose, Intrinsic universality of a 1-dimensional reversible cellular automaton, in: Symposium on Theoretical Aspects of Computer Science, 1997.

[9] E. Formenti, Cellular automata and chaos: from topology to kolmogorov complexity, Ph.D. thesis, École Normale Suprieure de Lyon (1998).

[10] R. H. Gilman, Classes of linear automata, Ergodic Theory and Dynamical Systems 7 (1) (1987) 105–118.

[11] E. Goles, A. Maass, S. Martinez, On the limit set of some universal cellular automata, Theoretical computer science 110 (1) (1993) 53.

[12] D. Griffeath, C. Moore, Life without death is **P**-complete, Complex Systems 10 (6) (1996) 437–447.

[13] G. A. Hedlund, Endormorphisms and automorphisms of the shift dynamical system, Mathematical Systems Theory 3 (1969) 320–375.

[14] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley, 1979.

[15] L. P. Hurd, Formal language characterizations of cellular automaton limit sets, Complex Systems 1 (1987) 69–80.

[16] K. Imai, K. Morita, Firing squad synchronization problem in reversible cellular automata, Theoretical Computer Science 165.

[17] J. Kari, The Nilpotency Problem of One-dimensional Cellular Automata, SIAM Journal on Computing 21 (1992) 571–586.

[18] P. Kůrka, Languages, equicontinuity and attractors in cellular automata, Ergodic Theory and Dynamical Systems 17 (1997) 417–433.

[19] P. Kůrka, Topological and symbolic dynamics, Socit Mathmatique de France, 2003.

[20] D. Lind, B. Marcus, An introduction to symbolic dynamics and coding, Cambridge University Press, Cambridge, 1995.

[21] J. Mazoyer, I. Rapaport, Additive cellular automata over Z_p and the bottom of $(\mathrm{ca}, \leq)$, in: Mathematical foundations of computer science (Brno, 1998), Springer, Berlin, 1998, pp. 834–843.

[22] J. Mazoyer, I. Rapaport, Global fixed point attractors of circular cellular automata and periodic tilings of the plane: undecidability results, Discrete Mathematics 199 (1999) 103–122.

[23] J. Mazoyer, I. Rapaport, Inducing an order on cellular automata by a grouping operation, Discrete Applied Mathematics 91 (1-3) (1999) 177–196.

[24] E. F. Moore, Machine Models of Self-Reproduction, in: Proceedings of Symposia in Applied Mathematics, vol. 14, American Mathematical Society, 1962.

[25] J. Myhill, The Converse of Moore's Garden-of-Eden Theorem, in: Proceedings of the American Mathematical Society, vol. 14, American Mathematical Society, 1963.

[26] T. Neary, D. Woods, P-completeness of cellular automaton rule 110, in: ICALP (1), 2006.

[27] N. Ollinger, Two-states bilinear intrinsically universal cellular automata, in: Fundamentals of Computation Theory, 13th International Symposium, FCT 2001, vol. 2138, Lecture Notes in Computer Science, 2001.

[28] N. Ollinger, The intrinsic universality problem of one-dimensional cellular automata, in: Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, 2003.

[29] N. Ollinger, Universalities in cellular automata: a (short) survey, in: B. Durand (ed.), Symposium on Cellular Automata Journées Automates Cellulaires (JAC'08), MCCME Publishing House, Moscow, 2008.

[30] M. Sablik, Directional dynamics for cellular automata: a sensitivity to initial condition approach, Theoretical Computer Science 400 (2008) 1–18.

[31] A. R. Smith, III, Simple computation-universal cellular spaces, Journal of the ACM 18 (1971) 339–353.

[32] K. Sutner, Cellular automata and intermediate degrees, Theor. Comput. Sci. 2 (296) (2003) 365–375.

[33] G. Theyssier, Captive cellular automata, in: MFCS, 2004.

[34] G. Theyssier, How common can be universality for cellular automata?, in: STACS, 2005.

[35] B. Weiss, Subshifts of finite type and sofic systems, Monatshefte für Mathematik 77 (1973) 462–474.

[36] S. Wolfram, Computation theory of cellular automata, Communications in Mathematical Physics 96 (1) (1984) 15–57.

A.3 Universalities in Cellular Automata

Version finale de [C5], un état de l'art sur les différentes formes d'universalité dans les automates cellulaires présenté à JAC'2008. Une version longue, plus complète, est à en cours d'écriture et à paraître dans [L1], un chapitre du *Handbook of Natural Computing*.

Journées Automates Cellulaires 2008 (Uzès), pp. 102-118

UNIVERSALITIES IN CELLULAR AUTOMATA A (SHORT) SURVEY

NICOLAS OLLINGER [1]

[1] Laboratoire d'informatique fondamentale de Marseille (LIF)
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France
E-mail address: Nicolas.Ollinger@lif.univ-mrs.fr

Abstract. This reading guide aims to provide the reader with an easy access to the study of universality in the field of cellular automata. To fulfill this goal, the approach taken here is organized in three parts: a detailed chronology of seminal papers, a discussion of the definition and main properties of universal cellular automata, and a broad bibliography.

Introduction

The idea and construction of a universal cellular automaton is as old as the formal study of the object itself, starting with the work of von Neumann [82] on self-reproduction in the 1940s, using cellular automata under suggestions by Ulam. Following the work of Turing, a Turing-universal cellular automaton is an automaton encompassing the whole computational power of the class of Turing machines, or by so-called Church-Turing thesis the class of recursive functions. To encode complex behaviors in a cellular automaton's dynamics, one can describe how to encode any computing device of a universal class of machines (Turing machine, tag systems, etc) and use classical tools of computability theory to shape wanted behaviors of the object. This is basically what von Neumann did. He designed a cellular automaton able to encode any Turing machine, the machine being moreover equipped with a construction arm controlled by the machine's head.

But Turing-universality is not the only reasonable kind of universality one might expect from cellular automata. It is quite unnatural to consider a universality of highly parallel potentially infinite devices as cellular automata by simulation of the dynamics of sequential finite machines — indeed, as we will discuss, to give a both widely acceptable yet precise definition of Turing-universality is a very difficult and unfulfilled challenge. As the study of cellular automata shifted both to dimension 1 and to the study of its dynamics, another kind of universality emerged. An intrinsically universal cellular automaton is an automaton able to properly simulate the behavior of any other cellular automaton on any type of

2000 ACM Subject Classification: F.1.1.
Key words and phrases: cellular automata, universality, reversibility.
This research has been supported by the French national research agency (ANR) grant Sycomore.

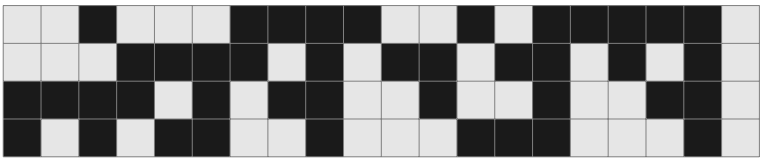


Figure 1: Partial space-time diagram of the $(\mathbb{Z}_2, +)$ rule

configuration (might it be infinite). It turns out that most of the historical constructions in dimension 2 and more, whereas designed as Turing-universal are Intrinsically Universal by the simple fact that they are designed to encode any boolean circuit.

A formal definition of universality might not seem so important. In fact, when building a precise cellular automaton from scratch to be universal, a definition is often implicit: the obtained behavior is the one engineered by the designer. The definition turns out to be more required when proceeding by analysis: given a cellular automaton rule, is it universal?

The present reading guide is constructed as follows. Section 1 *Cellular Automata* (p. 103) gives the definitions and notations used for cellular automata, configurations, and dynamics. Section 2 *Chronology* (p. 105) is an annotated chronology of seminal papers preparing to and concerning universality and universal cellular automata. Section 3 *Towards Formal Definitions* (p. 108) discusses the right definition of universalities in cellular automata. Section 4 *Higher Dimensions* (p. 110) discusses the construction and analysis of universal cellular automata in dimensions 2 and more, mostly using boolean circuits simulation. Section 5 *Turing Universality* (p. 111) discusses Turing universality, its links with universal Turing machines and the main technics of construction. Section 6 *Intrinsic Universality* (p. 113) discusses Intrinsic universality and the main technics of construction. Section 7 *Reversiblity and Universality* (p. 115) discusses universality in the special restricted case of reversible cellular automata.

1. Cellular Automata

A *cellular automaton* $\mathcal{A}$ is a tuple (d, S, N, f) where d is the *dimension* of space, S is a finite set of *states*, N a finite subset of $\mathbb{Z}^d$ is the *neighborhood* and $f : S^N \to S$ is the *local rule*, or *transition function*, of the automaton. A *configuration* of a cellular automaton is a coloring of the space by S, an element of $S^{\mathbb{Z}^d}$. The *global rule* $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ of a cellular automaton maps a configuration $c \in S^{\mathbb{Z}^d}$ to the configuration $G(c)$ obtained by applying f uniformly in each cell: for all position $z \in \mathbb{Z}^d$, $G(c)(z) = f(c(z + \nu_1), \dots, c(z + \nu_k))$ where $N = \{\nu_1, \dots, \nu_k\}$. A *space-time diagram* of a given cellular automaton is a mapping $\Delta \in S^{\mathbb{N}\times\mathbb{Z}^d}$ such that for all time step $t \in \mathbb{N}$, $\Delta(t + 1) = G(\Delta(t))$.

Example 1.1. Fig. 1 is a partial representation of a space-time diagram of the cellular automaton $(1, \mathbb{Z}_2, \{0, 1\}, f)$ where $f(x, y) = x + y$. State 0 is represented by the white color, state 1 by the black color. Time goes from bottom to top.

In this paper, we consider for the most part cellular automata of dimension 1 and 2 with the typical neighborhoods depicted on Fig. 2: von Neumann $\{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ and Moore $\{-1, 0, 1\}^2$ in dimension 2, first neighbors $\{-1, 0, 1\}$ and one way $\{-1, 0\}$ in dimension 1.

(a) von Neumann (b) Moore (c) first neighbors (d) one way

Figure 2: Typical neighborhoods

Several subsets of the space of configurations are considered. Given a *quiescent state* q satisfying $f(q,\dots,q) = q$, a *q-finite configuration* c is a configuration equal to q in all but finitely many cells: there exists α such that for all position $z \in \mathbb{Z}^d$, $\|z\|_\infty > \alpha \to c(z) = q$. A configuration c admits p as a *periodicity vector* if for all position $z \in \mathbb{Z}^d$, $c(z+p) = c(z)$. A configuration c in dimension d is *periodic* if it admits a family of d non-colinear periodicity vectors: there exists $p \in \mathbb{N}^d$ such that $(p_1, 0, \dots, 0)$, $(0, p_2, 0, \dots, 0)$, $\dots$, and $(0, \dots, 0, p_d)$ are periodicity vectors of c. A configuration c in dimension d is *ultimately periodic* if there exists α and d non-colinear vectors v_i such that for all position $z \in \mathbb{Z}^d$ and all vector v_i, $\|z\|_\infty > \alpha \to c(z + v_i) = c(z)$. Notice that in dimension 1, an ultimately periodic configuration can have two different ultimately periodic pattern on each side.

Constraints can also be added to the local rule. Symmetries are usually considered to obtain more natural rules mimicking physical systems. A symmetry rule can be seen as a one-to-one mapping $\rho : \mathbb{Z}^d \to \mathbb{Z}^d$: the image of a configuration $c \in S^{\mathbb{Z}^d}$ by the symmetry rule ρ is the configuration $\rho(c)$ satisfying for all position $z \in \mathbb{Z}^d$, $\rho(c)(z) = c(\rho(z))$. A cellular automaton $\mathcal{A}$ respects a symmetry rule ρ if ρ and G commute, *i.e.* $\rho(G(c)) = G(\rho(c))$. Typical symmetries include reflections around point ($\rho_0(x,y) = (-x,-y)$), around axes ($\rho_x(x,y) = (-x,y)$) and rotations ($\theta(x,y) = (-y,x)$). A cellular automaton is *totalistic* if it's set of states is a subset of $\mathbb{N}$ and the local rule f can be written as $f(s_1,\dots,s_k) = g(\sum_{i=1}^k s_i)$. Totalistic rules respect all symmetries that preserve the neighborhood (*i.e.* such that the image of the neighborhood by the symmetry rule is equal to the neighborhood): totalistic cellular automata with the von Neumann or Moore neighborhood are reflection and rotation invariants.

A cellular automaton is injective (resp. surjective, one-to-one) if its global rule is injective (resp. surjective, one-to-one). A cellular automaton $\mathcal{A}$ is reversible if there exists a cellular automaton $\mathcal{B}$ that reverts it, that is such that $G_\mathcal{B} \circ G_\mathcal{A}$ is the identity map.

Proposition 1.2 (Hedlund [31], Richardson [72])**.** *A cellular automaton is reversible if and only if it is injective.*

Proposition 1.3 (Amoroso and Patt [2])**.** *It is decidable given a one-dimensional cellular automaton to decide whether it is reversible.*

Proposition 1.4 (Kari [34, 35])**.** *It is undecidable given a two-dimensional cellular automaton to decide whether it is reversible.*

Whereas reversibility is an undecidable question, the construction of reversible cellular automata is possible, provided that the backward rule is constructed at the same time as the forward rule. Partitioned cellular automata provide a convenient way to construct reversible cellular automata. A *partitioned cellular automaton* is a cellular automaton with state set

$S_1 \times S_2 \times \cdots \times S_k$ whose local rule can be rewritten as $f((s_1^1, \ldots, s_1^k), \ldots, (s_k^1, \ldots, s_k^k)) = \varphi(s_1^1, s_2^2, \ldots, s_k^k)$ where $\varphi : \prod S_i \to \prod S_i$ is the partitioned rule. As it is straightforward to verify, a partitioned cellular automaton is reversible if and only if its partitioned rule is one-to-one. As the partitioned rule is a mapping from a finite set to itself, any partially defined injective rule can be completed to a reversible cellular automaton.

For a better and more complete introduction to the theory of cellular automata, see Delorme [18] and/or Kari [36].

2. Chronology

It is a difficult task to give a fair and complete chronology of a research topic. In this section, we propose an exploration of the history of the field in three main eras:

(1) the *computation and machines* era describes seminal papers outside the realm of cellular automata that lead to the main tools necessary to consider computation in the context of abstract machines;
(2) the *universality and cellular automata* era is the core part of the chronology: it describes seminal papers along the path of universality study in the realm of cellular automata, from the early work of von Neumann in the 50s to the end of the 90s;
(3) the *recent trends* era is a more subjective choice of some papers in the field in the twenty-first century.

2.1. Computation and Machines

Gödel 1931 [30]**:** in his now classical paper describing incompleteness theorems, Gödel introduces so-called Gödel numberings: the ability to encode and manipulate a formal system inside itself if the system is complex enough. The concept of universality directly depends on such an encoding: a universal machine simulates a machine through its encoding. For a precise analysis from a logic and computer science point of view of Gödel's paper, see Lafitte [38].

Turing 1936 [81]**:** while introducing Turing machines and proving the undecidability of the halting problem by a diagonal argument, Turing also introduces its universal machine. Fixing an enumeration of Turing machines and a recursive bijective pairing function $\langle ., . \rangle : \mathbb{N}^2 \to \mathbb{N}$, he describes a machine U that, on input $\langle m, n \rangle$, computes the same value as the machine encoded m on input n. Universality as a property is not discussed: a unique universal Turing machine U is given. For a discussion of the development of ideas from Leibniz to Turing results, see Davis [17].

Post 1943 [68]**:** at that time, many different models of computation where proposed and proved equivalent, leading to the so-called Church-Turing thesis. Post introduces tag systems, a combinatorial word-based system successfully used since to construct size-efficient universal Turing machines. For a modern definition and discussion of tag systems, see Minsky [51].

Kleene 1956 [37]**:** finite state machines are at the hearth of many models of computation. Kleene's paper proves the equivalence between three different families of objects: regular languages, finite automata, and boolean circuits. Boolean circuits are modeled after the formal study of abstract neurons by McCulloch and Pitts [49]. This equivalence is fundamental both to concrete computer design and discrete models of computation like cellular automata. For a modern discussion on

this equivalence and its consequences from the point of view of computation and machines, see Minsky [51]. Perrin [66] gives an history of this period and important achievements with respect to the field of finite automata and formal languages.

Minsky 1967 [51]**:** In the spirit of the question from Shannon [73] about the size of a smallest Turing machine, Minsky explains how to efficiently encode tag systems computations into Turing machines and describe a universal Turing machine with four symbols and seven states. This marks the real start of a (still running) competition.

Lecerf 1963 [40]**, Bennett 1973** [8]**:** Reversible computation is concerned with computing devices that can unroll their computation, going back in time. In their independent papers, Lecerf and Bennett prove that reversible Turing machines are able to simulate just any Turing machine. Thus, there exists reversible Universal Turing machines.

Fredkin and Toffoli 1982 [27]**:** To encode classical computations into discrete models, Kleene [37]'s theorem permits to go freely from circuits to finite state machine, an essential ingredient for computation. Fredkin and Toffoli discuss an analogous for reversible computation: elementary building blocks to encode any reversible finite state machine as a reversible circuit. This paper also introduces the so-called billard ball model of computation: a discrete cellular automata model to encode reversible computations. The encoding of reversible finite state machines into circuits was later improved by Morita [55].

2.2. Universality and Cellular Automata

von Neumann 1966 [82]**:** Introducing cellular automata in order to construct a self-reproducing machine, participating to the reflexion on the nature of life, von Neumann takes a fixed-point approach. His two-dimensional, 29 states, von Neumann neighborhood cellular automaton is able to simulate a particular kind of Turing machine that can also control a construction arm. The power of the construction arm is rich enough to construct with finitely many instructions a copy of the Turing machine itself. Whereas the machine is constructed with a form of Turing-universality in mind, the simulation of the Turing machine is done with very simple components wiring down a particular family of boolean circuits. As a consequence, the original cellular automaton is also, *a posteriori*, intrinsically universal. The construction of von Neumann leads to various improvements and discussions on the encoding of boolean circuits, the different organs that compose the machine and the transmission of signals. A non exhaustive list of interesting following papers might be: Arbib [3], Burks [10, 11, 12], Moore [52, 53], Thatcher [77, 76].

Codd 1968 [14]**:** Following the principle of von Neumann idea on self-reproduction, Codd drastically reduces the complexity of the automaton. Codd's two-dimensional rule uses 8 states with the von Neumann neighboorhood. Signals are conveyed by pairs of states (an oriented particle) moving between walls and reacting upon collision. This cellular automaton is also universal for boolean circuits and so intrinsically universal. A latter construction by Langton [39], based on Codd ideas, has fewer states and a very simple family of self-reproducing loops but looses its computation universal capabilities.

Banks 1970 [4, 5]**:** The work of Banks is noticeable with respect to several aspects and also because of its relatively small diffusion in the cellular automata community. Banks constructs a family of very small cellular automata (two-dimensional, von Neumann neighborhood, very symmetric, four to two states) simulating boolean circuits in a very simple and modern way (signals moving in wires, boolean gates on collisions), he identified and used explicitly the property of intrinsic universality and gave a transformation to construct relatively small universal one-dimensional cellular automata with large neighborhoods starting from two-dimensional ones (reencoding it into a one-dimensional first-neighbors automaton with 18 states). Construction of a two-dimensional four state universal cellular automaton in the spirit of Banks is provided by Noural and Kashef [61].

Conway 1970 [29, 9]**:** The Game of Life introduced by Conway is certainly among the most famous cellular automata and the first rule to be proven universal by analysis of a given rule rather than on purpose construction. A modern exposition of the Game of Life universality and a proof of its intrinsic universality was latter proposed by Durand and Róka [22].

Smith III 1971 [74]**:** The simulation of Turing machine by cellular automata to construct one-dimensional Turing-universal cellular automata is studied by Smith III. Among several results, he explains how to construct a one-dimensional Turing-universal cellular automaton with first neighbors and 18 states.

Toffoli 1977 [80]**:** Any cellular automaton of dimension d can be simulated, in a certain sense, by a cellular automaton of dimension $d+1$. Using this assertion, Toffoli shows that two-dimensional reversible cellular automata can be Turing-universal. The result was later improved by Hertling [32].

Margolus 1984 [42]**:** Whereas Toffoli transforms any Turing machine into a two-dimensional cellular automaton by using a new spatial dimension to store computational choices, Margolus constructs a Turing-universal two-dimensional reversible cellular automaton by simulation a bouncing billard ball, complex enough to compute any reversible boolean function of conservative logic. The billard ball model cellular automaton has 16 states defined as two-by-two blocks of binary cells and von Neumann neighborhood.

Albert and Čulik 1987 [1]**:** Each cellular automaton can be simulated by a totalistic cellular automaton with one-way neigborhood. With the help of the last proposition, Albert and Čulik construct the first universal cellular automaton obtained by simulation of any cellular automaton of the same dimension. The automaton works along the following principle: each macro-cell copies the state of its left neighbor and adds it to its state obtaining some n, then by copying the nth element of a reference table, it selects its new state. Whereas the spirit of intrinsic universality is definitely there, the technical implementation is less clear. The one-dimensional first-neighbors automaton obtained has 14 states. The construction was later improved by Martin [43, 44] with better transition time complexity and smn theorem.

Morita and Harao 1989 [57]**:** Introducing partitioned cellular automata, Morita and Harao explicitely simulate any reversible Turing machine on a one-dimensional reversible cellular automaton, proving that one-dimensional reversible cellular automata can be Turing-universal. The construction was later improved by Dubacq [21], simulating any Turing machine in real time (without loss of time).

Lindgren and Nordahl 1990 [41]**:** The direct simulation of Turing machine on one-dimensional cellular automata proposed by Smith III can be improved and any m states n symbols machine can be simulated by a $(m + n + 2)$-states cellular automaton following Lindgren and Nordhal. Applying this to Minsky's 7 states and 4 symbols machine and then transforming the simple simulation into a macro-state signal based simulation, Lindgren and Nordhal obtain a one-dimensional first neighbors 7 state Turing-universal cellular automaton. The intrinsic universality status of this automaton is unknown.

Durand and Róka 1996 [22]**:** Revisiting the Game of Life and filling holes in the universality proof, Durand and Róka publish the first discussion on the different kinds of universality for cellular automata and the problem of formal definition.

Durand-Lose 1997 [25]**:** Using a modern definition of intrinsic universality, Durand-Lose goes one step further than Morita and Harao by constructing a one-dimensional cellular automata intrinsically simulating any cellular automaton.

2.3. Recent Trends

Imai and Morita 2000 [33]**:** The improvement in the construction of small and simple two-dimensional reversible cellular automata continues. Imai and Morita use partitioned cellular automata to define an 8 state universal automaton.

Ollinger 2002 [64]**:** Using simulation technics between cellular automata, strong intrinsically universal cellular automata with few states can be constructed, here 6 states.

Cook 2004 [15]**:** Very small universal cellular automata cannot be constructed, they have to be obtained by analysis. Realising a real tour de force, Cook was able to prove the Turing-universality of the 2-states first-neighbors so-called rule 110 by analysing signals generated by the rule and their collisions. The intrinsic universality of this automaton remains open. The original construction, simulation of a variant of tag system, was exponentially slow. For a proof of Cook's result using signals, see Richard [70].

Neary and Woods 2006 [60]**:** Recently, the prediction problem of rule 110 was proven P-complete by Neary and Woods by a careful analysis and modification of Turing machines simulation technics by tag systems. As P-completeness is required for intrinsic universality, this is another hint of the potential strong universality of rule 110.

Richard 2008 [71]**:** The limits of constructed small intrinsically universal cellular automata are converging towards analysed cellular automata. Using particles and collisions, Richard was recently able to construct a 4 state intrinsically universal first-neighbors one-dimensional cellular automaton.

3. Towards Formal Definitions

What is a universal cellular automaton? At first, the question might seem simple and superficial: a universal cellular automaton is an automaton able to compute anything recursive. In fact, a formal definition is both required and difficult to obtain. The requirement for such a definition is needed to define a frontier between cellular automata of maximal complexity and the others: in particular when considering the simplest cellular automata,

to be able to identify the most complex. The difficulty arise from the fact that we both want a definition broad enough to encapsulate all constructions of the literature and all *fair enough* future constructions. For more details concerning this philosophical question, see Durand and Róka [22] attempt to give formal definitions.

Turing-universality is the easiest form of universality one might think about, that is with a computability culture: let the cellular automaton *simulate* a well known universal model of computation, either simulating one universal object of the family or any object of the family.

The first approach pushes back the problem to the following one: what is a universal Turing machine? a universal tag system? In its original work, Turing did not define universal machines but *a* unique universal machine. The definition of universality for Turing machines was later discussed by Davis [16] who proposed to rely on recursive degrees, defining universal machines as machines with maximal recursive degree. This definition while formal lacks precise practical view of encoding problems: the issue continues to be discussed in the world of Turing machines, becoming more important as smaller and smaller universal machines are proposed. For a view on the universality of Turing machines and pointers to literature related to the topic, see Woods [86].

The second approach leads to the problem of heterogeneous simulation: classical models of computation have inputs, step function, halting condition and output. Cellular automata have no halting condition and no output. As pointed out by Durand and Róka [22], this leads to very tricky encoding problems: their own attempt of a Turing-universality based on this criterium as encoding flaw permitting counter-intuitively to consider very simple cellular automata as universal.

Turing-universality of dynamical systems in general and cellular automata in particular has been further discussed by Delvenne, Kůrka, and Blondel [19], and Sutner [75]. None of the proposed definition are completely convincing so forth, so we will choose on purpose not to provide the reader with yet another weak formal definition.

Intrinsic universality, on the other hand, is easier to formalize, yet more robust notion (in the sense that variations along the lines of the definition lead to the same set of universal automata). Consider a homogenous type of simulation: cellular automata simulated by cellular automata in a shift invariant, time invariant way. A natural type of universal object exist in this context: cellular automata able to simulate each cellular automaton. Following the ideas of grouping and bulking [69, 47, 63], we introduce a general notion of simulation broad enough to scope all reasonable constructions of the literature.

Direct simulation between two cellular automata can be formalized as follows. A cellular automaton $\mathcal{B}$ *directly simulates* a cellular automaton $\mathcal{A}$, denoted $G_{\mathcal{A}} \prec G_{\mathcal{B}}$, of the same dimension according to a mapping $\varphi : S_{\mathcal{A}} \to 2^{S_{\mathcal{B}}}$ if for any pair of states $a, b \in S_{\mathcal{A}}$, $\varphi(a) \cap \varphi(b) = \emptyset$ and for any configuration $c \in S_{\mathcal{A}}^{\mathbb{Z}^d}$, $G_{\mathcal{B}}(\varphi(c)) \subseteq \varphi(G_{\mathcal{A}}(c))$.

For any state set S, let $(m_1, \dots, m_d)$ be a tuple positive integers, the *unpacking bijective map* $o_{(m_1,\dots,m_d)} : \left(S^{\prod m_i}\right)^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is defined for any configuration $c \in \left(S^{\prod m_i}\right)^{\mathbb{Z}^d}$ and any position $z \in \mathbb{Z}^d$ and $r \in \prod_i \mathbb{Z}_{m_i}$ as $o_{(m_1,\dots,m_d)}(c)(m_1 z_1 + r_1, \dots, m_d z_d + r_d) = c(z)(r)$. The *translation* of vector $v \in \mathbb{Z}^d$ is defined for any configuration $c \in S^{\mathbb{Z}^d}$ and position $z \in \mathbb{Z}^d$ as $\sigma_v(c)(z) = c(z - v)$.

Simulation between two cellular automata is extended by considering packing, cutting and shifting of the two cellular automata such that direct simulation occur between both transformed objects. Universal objects are then maximum of the induced pre-order. In fact, it can be proved that simulation on one side is sufficient for universal objects.

Definition 3.1 (intrinsic universality)**.** A cellular automaton $\mathcal{U}$ is intrinsically universal if for each cellular automaton $\mathcal{A}$ of the same dimension there exists an unpacking map o_m, a positive integer $n \in \mathbb{N}$ and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Proposition 3.2 (Mazoyer and Rapaport [69, 47])**.** *No cellular automaton is intrinsically universal in real time (that is, when constraining cutting constant n to be equal to* $\max(m)$*): simulation cannot perform both information displacement and transition computation at the same time.*

Proposition 3.3 (Ollinger [65])**.** *Given a cellular automaton, it is undecidable to determine whether it is intrinsically universal.*

Turing-universality and intrinsic universality notions are really different notions. Some erroneous claims by Wolfram [83, 84] affirm for example that rule 110 is intrinsically universal. In fact, the question is yet open, Turing universality is the only proven thing.

Proposition 3.4 (Ollinger [63], Theyssier [78])**.** *There exists Turing-universal cellular automata which are not intrinsically universal. Moreover, some of them are at the bottom of an infinite increasing chain of equivalences classes of the preorder.*

Universality can also be discussed when considering language recognition or computation on grids. This topic is out of scope of the present paper. For more on this topic, see Mazoyer [45, 46].

4. Higher Dimensions

In two and more dimensions, an easy way to construct both intrinsically and Turing universal cellular automata is to go through boolean circuit simulation. Boolean circuits can encode any finite state machine and a cell of a cellular automaton or the control and tape of a Turing machine can be described as finite state machines. The topic of boolean circuit simulation with cellular automata is quite popular and a lot has been written on it, see for example recreations around the wireworld cellular automaton designed by Silverman and discussed in Dewdney [20]. Let us just here give technical hints and possible exotic extensions without entering details.

To simulate boolean circuits, one typically needs to mix the following ingredients:

wires: boolean signals travel in piecewise straight line in space, their paths are the wires. Several encoding of boolean signals with or without explicit wires are possible: moving particles encoded as states with a direction vector, bouncing on walls to turn as in game of life [29]; wire path encoded as wire cells with explicit direction vector on each wire cell as in von Neumann [82]; undirected wire cells on which directed signals travel; undirected wire cells on which pairs of two different signal cells travel, the direction being given by the orientation of the pair as in wireworld [20]; pairs of undirected wire paths in between which pairs of two different signal cells travel as in Codd [14] or Banks [5].

turn and delay: boolean signals should be able to turn in space and delay their arrival to permit signal synchronization.

signal crossing: in order to encode all boolean circuits, crossing of signals has to be encoded either explicitly (adding crossing states) or implicitly using delaying technics (as in von Neumann [82]) or boolean logic tricks.

gates: signals must be combined using boolean gates at least taken in a boolean universal family of gates. AND, OR, NOT is the classical one but NAND or NOR is sufficient alone.

fan-out: signals must be duplicated in some way either with an explicit fan-out state or using specific wire split rules.

Remarks and encoding tricks regarding boolean circuit simulation:

- Universal boolean functions families and their expressive power are described in Post [67]. But, in cellular automata encoding, it is easy to use constants and multiple wires encoding, thus the number of boolean classes depending on the implemented gates is finite and small.
- Clocks are only needed when dealing with some form of synchronized logic simulation. It is often used because boolean signals are encoded with two values: empty wire or signal on wire. With such an encoding, NOT gate has to generate new signal on wire and clock signal is used to determine at which time steps to do so. However, a classical coding trick to avoid the use of clocks and diods is to only implement OR and AND gates and use the two wires trick to gain boolean universality: a signal is encoded as one signal on one wire, the second being empty (thus no signal is encoded as no signal on both wires), then the NOT gate is just the wire crossing $(x, y) \mapsto (y, x)$, the AND gate can be encoded as $(x, y) \mapsto (x \wedge y, x \vee y)$ and the OR gate as $(x, y) \mapsto (x \vee y, x \wedge y)$. As both OR and AND produce signal only if there is at least one signal in input, the need for clock vanishes.
- Wire crossing can be gained for free by using the XOR gate as planar crossing can be implemented with XORs.
- Delays come for free if the wires can turn in all directions.
- In dimension 3, wire crossing is not needed, use the third dimension to route wires.
- Signal encoding can be done using signal constructions, in the spirit of Mazoyer and Terrier [48], in order to reduce the number of states.

Of course, boolean circuit simulation is not restricted to square grids. As an example of a more exotic lattice, Gajardo and Goles [28] encoded a boolean circuit simulator on a hexagonal lattice (with proper cellular automata definition).

Small intrinsically universal cellular automata are quite simple to construct in dimension two with few states: Banks does it with 2 states and von Neumann neighborhood with reflection and rotation symmetry.

5. Turing Universality

In dimension one, boolean circuit encoding is more puzzling as wire crossing capabilities is bounded by the local rule. Thus, historically, computation universality is achieved by direct simulation of universal models of computations :

Turing machines: Turing machines are easy to encode on cellular automata (see below) as an infinite tape really looks like a configuration of a cellular automaton.

In fact, several variants of Turing machines exist and an important literature on universality in the Turing world provide useful objects to build small universal automaton based on this model. The question of existence of small universal Turing machines was first raised by Shannon [73], different variants of Turing machines are discussed by Fischer [26]. For a survey on small Turing machine construction, see Woods and Neary [86].

Tag systems: Tag systems provide a better model to design very small universal objects. In fact, very small universal Turing machines are constructed by simulation of tag systems and their variants as originally proposed by Minsky [51, 13]. The original drawback of tag system was its exponential slow-down when simulating Turing machines, this drawback was removed recently by Woods and Neary [85, 86] achieving polynomial time simulation. The Turing-universality of rule 110 is obtained by Cook [15] by direct simulation of a proper variant of tag systems.

The variant of Turing machine we use is the following. A *Turing machine* is a tuple (S, Σ, B, s_0, T) where S is a finite set of states, Σ is a finite alphabet with a special blank symbol $B \in \Sigma$, $s_0 \in S$ is the initial state and $T : S \times \Sigma \to S \times \Sigma \times \{\leftarrow, \rightarrow\}$ is a partial transition map. A transition rule $T(s, a) = (s', b, d)$ reads as follow: when reading a from state s, write b on the tape, move in direction d, and enter state s'. A configuration of the machine is a triple (s, z, c) where $s \in S$ is the current state of the machine, $z \in \mathbb{Z}$ is the position of the head, and $c \in S^{\mathbb{Z}}$ is the content of the tape. The machines goes in one step from a configuration (s, z, c) to a configuration (s', z', c') if the transition rule $T(s, c(z)) = (s'', d, b)$ is defined and verifies $s' = s''$, $z' - z = d$, $c'(z) = b$ and for all position $z'' \neq z$, $c'(z) = c(z)$. Starting from a configuration $\mathfrak{c}$, an halting computation of the machine in time t consists of a sequence of configurations $(\mathfrak{c}_i)_{i=0}^{t}$ such that $\mathfrak{c}_0 = \mathfrak{c}$, the machine cannot reach any configuration from $\mathfrak{c}_t$ and for all i, the machine goes in one step from $\mathfrak{c}_i$ to $\mathfrak{c}_{i+1}$. The configuration $\mathfrak{c}_t$ is the output of the computation.

Following Smith III [74], a given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows. Let $S' = \Sigma \cup S \times \Sigma$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: $c'(z) = (s, c(z))$ and for all positions $z' \neq z$, $c'(z') = c(z)$. The local rule encodes the transition function of the Turing machine. For each transition $T(s, a) = (s', b, \leftarrow)$, for all states $x, y \in S$, $f(x, y, (s, a)) = (s', y)$ and $f(x, (s, a), y) = b$. Symmetrically, for each transition $T(s, a) = (s', b, \rightarrow)$, for all states $x, y \in S$, $f((s, a), y, x) = (s', y)$ and $f(x, (s, a), y) = b$. All undefined transitions apply identity: $f(x, y, z) = y$. With this encoding, starting from an encoded configuration $\tau(\mathfrak{c})$, the configuration evolves in one step to a configuration $\tau(\mathfrak{c}')$ where $\mathfrak{c}'$ is the next computation step of the Turing machine if it exists, $\mathfrak{c}' = \mathfrak{c}$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $(m + 1)n$ states.

To lower the number of states, Lindgren and Nordahl [41] introduce a simulation scheme where each step of the Turing machine computation is emulated by two time steps in the cellular automaton. A given Turing machine (S, Σ, B, s_0, T) can be simulated by a cellular automaton $(1, S', \{-1, 0, 1\}, f)$ as follows. Let $S' = \Sigma \cup S \cup \{\bullet, \leftrightarrow\}$. A configuration (s, z, c) of the Turing machine is encoded as a configuration $c' = \tau(s, z, c)$ of the cellular automaton in the following way: for all $z' < z$, $c'(2z') = \bullet$, $c'(2z' + 1) = c(z)$; for all $z' > z$, $c'(2z' + 1) = \bullet$, $c'(2z' + 2) = c(z)$; $c'(2z) = \bullet$ and either $c'(2z + 1) = s$, $c'(2z + 2) = c(z)$

or $c'(2z+1) = c(z)$, $c'(2z+2) = s$ (two possible encodings). The local rule encode the transition function of the Turing machine. Applying the rule: for each transition $T(s,a) = (s', b, \leftarrow)$, $f(\bullet, s, a) = s'$, $f(s, a, \bullet) = b$, $f(\bullet, a, s) = s'$, $f(a, s, \bullet) = b$, for each transition $T(s,a) = (s', b, \rightarrow)$, $f(\bullet, s, a) = b$, $f(s, a, \bullet) = s'$, $f(\bullet, a, s) = b$, $f(a, s, \bullet) = s'$. Moving: for all $s \in S$, $a, b \in \Sigma$, $f(\leftrightarrow, s, a) = \bullet$, $f(a, \leftrightarrow, s) = s$ $f(a, s, \leftrightarrow) = \bullet$, $f(s, \leftrightarrow, a) = s$. All undefined transitions apply the identity rule but for $\bullet$ and $\leftrightarrow$ that alternates: for all states $x, y \in S$, $f(x, \bullet, y) = \leftrightarrow$ and $f(x, \leftrightarrow, y) = \bullet$ With this encoding, starting from an encoded configuration $\tau(\mathfrak{c})$, the configuration evolves in two steps to a configuration $\tau(\mathfrak{c}')$ where $\mathfrak{c}'$ is the next computation step of the Turing machine if it exists, $\mathfrak{c}' = \mathfrak{c}$ otherwise. Using this simulation, a Turing machine with m states and n symbols is simulated by a one-dimensional cellular automaton with first-neighbors and $m + n + 2$ states.

Simulation of tag systems is more tricky due to the non-locality of one computation step of the system. Following Cook [15], one can consider cyclic tag systems. A cyclic tag system is given as a finite set of words $(w_0, \ldots, w_{N-1})$ on the alphabet $\{\circ, \bullet\}$. A configuration of the system is a word $u \in \{\circ, \bullet\}^*$. At time step t, the configuration u evolves to a configuraton v according if either $u_0 = \circ$ and $u_0 v = u$, either $u_0 = \bullet$ and $u_0 v = u w_{t \mod N}$. Cyclic tag systems can encode any recursive function. To encode all cyclic tag systems in a same cellular automaton $(1, S, \{-1, 0, 1\}, f)$, one can follow the following principle. Encode each configuration u of a cyclic tag system $(w_0, \ldots, w_{N-1})$ as a configuration of the kind $^\omega(T\_^k) \cdot u \cdot \blacksquare (w_0 \blacktriangle \ldots \blacktriangle w_{N-1})^\omega$ where intuitively T is a clock signal, $\blacksquare$ is the frontier between u and the rule and the rule is repeated on the right each word separated by a $\blacktriangle$. Giving the complete local rule is tedious but let us sketch its principle: each time a clock signal hits the first letter of u, it erases it and send a signal to the right transporting the value of that letter; when the signal meets the $\blacksquare$ it removes it and begins to treat the w word on the right, either erasing it or just crossing it; when the signal meets a $\blacktriangle$, it changes it into a $\blacksquare$ and the signal disappears. This principle is used by Cook to simulate cyclic tag systems with rule 110 particles and collisions.

A main point of discussion there is to decide which kind of configurations are acceptable for encoding Turing-universal computation. Finite configurations are certainly not a problem and using any, potentially non-recursive, configuration would permit trivial cellular automata to be misleadingly called universal. The previous constructions involving Turing machines use finite or ultimately periodic configurations with a same period on both sides, the same one for all simulated machines, whereas the tag system encoding uses ultimately periodic configurations with different periods, moreover these periodic parts depend on the simulated tag system. The tag system really needs this ultimate information, transforming the simulating cellular automaton into one working on finite configurations would have a constant but large impact on the number of states. As pointed in Durand and Róka [22], this configuration encoding problem adds difficulties to the formal definition of Turing-universality.

6. Intrinsic Universality

Even if the concept of intrinsically universal cellular automata took some time to emerge, intrinsic universality does not require more complex constructions to be achieved. Several technics are used to construct them:

Parallel Turing machines table lookup: A simple way to achieve intrinsic universality is to use synchronized parallel Turing heads (one copy of the same Turing machine per encoded cell) to lookup in the transition table (one copy in each encoded cell) of the encoded cellular automaton. Notice that the Turing machines used for this are not the same ones that are Turing-universal. In fact, their computational power is very small but they can carry precise information movement tasks.

One-way totalistic lookup: Another more cellular automata centric way to achieve intrinsic universality is, following Albert and Čulik 1987 [1], to simplify the task of the previous machine by simulating only one-way totalistic cellular automata which are sufficient to simulate all cellular automata.

Signals: The previous models are complex because the information displacement involved is still complex due to the sequential behavior of head-based machines. Following Ollinger [64] and Richard [71], particles and collisions, that is signals in the style of Mazoyer and Terrier [48], can be used to encode the information and perform the lookup task with parallel information displacement.

We explain here the parallel Turing machines table lookup technic, the other ones being refinements based on it. The one-dimensional first-neighbors universal cellular automaton $\mathcal{U}$ simulates a cellular automaton $(1, S, \{-1, 0, 1\}, f)$ the following way. Each configuration c is encoded as the concatenation of $\psi_{\mathcal{A}}(c(z))$ for all z. For each state $s \in S$, $\psi_{\mathcal{A}}(s)$ is a word of the kind $\blacksquare\tau(f(1,1,1)) \bullet \tau(f(1,1,2)) \bullet \ldots \bullet \tau(f(N,N,N))\blacktriangle 0^k\tau(s)0^k0^k$ where N is the size of S, k is the number of bits needed to encode numbers from 1 to N and $\tau(s)$ is a binary encoding of the state s. The simulation proceeds as follows so that the movement of each head is the same, up to translation, on each well encoded configuration. First the $\blacksquare$ letter is activated as a Turing head in initial state. The Turing head then moves to the left and copies the state $\tau(s_L)$ of the left neighbor in place of the first 0^k block. Then it symmetrically copies the state $\tau(s_R)$ of the right neighbor in place of the second 0^k block. This being done, the head scans the entire transition table, incrementing a counter (for example stored on top of the encoded states) at each step: if at some point the counter is equal to the triple of states, the result is copied from the transition table to the third 0^k block. At the end of the scan, the counter information is cleared, the result of the transition is copied in the $\tau(s)$ place and all three 0^k blocks are restored. The head then goes back to the $\blacksquare$. Using this simulation, one step of the simulated cellular automaton is simulated in a constant number of step for each cell by each Turing head. The universal automaton does not depend on the simulated automaton and is so intrinsically universal. A careful design can lead to less than 20 states. If the simulation uses the one-way totalistic technic, encoding states in unary, then it is easy to go under 10 states.

Notice that general Turing-universal cellular automata construction schemes from previous section concerning Turing machines can be adapted to produce small intrinsically universal cellular automata: apply the encoding schemes on machines performing the cell simulation task. However, the tag system simulations do not provide direct way to obtain intrinsic universality. Moreover, it is possible to design cellular automata Turing-universal by tag system simulation and not intrinsically universal.

More exotic intrinsically universal cellular automata have been studied on constrained rules. As an example, Moreira [54] constructed number conserving intrinsically universal automata, Bartlett and Garzon [6, 7] and Ollinger [62] do the same for bilinear cellular

automata, and Theyssier [79] for captive cellular automata for which he proves that almost all captive automata are intrinsically universal.

7. Reversibility and Universality

Reversible cellular automata are special in the sense that they can achieve Turing-universality as any Turing machine can be simulated by a reversible Turing machine but they cannot achieve intrinsic universality: reversible cellular automata only simulate reversible cellular automata. However, there exists reversible cellular automata which are universal with respect to the class of reversible cellular automata.

Definition 7.1 (reversible intrinsic universality)**.** A reversible cellular automaton $\mathcal{U}$ is intrinsically universal for reversible cellular automata if for each reversible cellular automaton $\mathcal{A}$ of the same dimension there exists an unpacking map o_m, a positive integer $n \in \mathbb{N}$ and a translation vector $v \in \mathbb{Z}^d$ such that $G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v$.

Turing-universality and weak form of intrinsic universality have been proposed by Morita [56], Morita and Imai [58, 59], Durand-Lose [23, 24], Miller and Fredkin [50]. As for classical cellular automata in higher dimension the simulation of reversible boolean circuits automatically gives reversible intrinsic universality.

For one-dimensional cellular automata, reversible intrinsic universality can be achieved by simulating any one-way reversible partitioned reversible cellular automaton with a first-neigbors reversible partitioned reversible cellular automaton. We briefly sketch how to use a scheme similar to parallel Turing machine table lookup with reversible Turing machines to achieve this goal. The first adaptation is to remark that a the local partition rule of a reversible automaton is a permutation, thus it can be encoded as a finite sequence of permutation pairs. So, the table of transition is encoded as a finite sequence of pairs of states. The reversible Turing machine task is to scan the transition table and for each pair it contains to replace the actual state by the second element of the pair if the state appears in the current pair. It is technical but straightforward to see that a reversible Turing machine can achieve this. The information movement is reversible as in partitioned automata each cell gives half its state to a neighbor and take half a state from the other without erasing any information. Developing this simulation scheme, one constructs a reversible intrinsically universal cellular automaton.

8. Conclusion

This brief reading guide has given the reader the keys both to further explore the literature and to construct by itself conceptually simple Turing-universal and/or intrinsically universal cellular automata in one and two dimensions. The following broad bibliography can be explored with the help of the main text, all references being cited.

References

1. J. Albert and K. Čulik, II, *A simple universal cellular automaton and its one-way and totalistic version*, Complex Systems **1** (1987), no. 1, 1–16.
2. S. Amoroso and Y. N. Patt, *Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures*, Journal of Computer and System Sciences **6** (1972), 448–464.
3. M. A. Arbib, *Simple self-reproducing universal automata*, Information and Control **9** (1966), no. 2, 177–189.
4. E. R. Banks, *Universality in cellular automata*, Symposium on Switching and Automata Theory (Santa Monica, California, 1970), IEEE, 1970, pp. 194–215.
5. ———, *Information processing and transmission in cellular automata*, Ph.D. thesis, Massachusetts Institute of Technology, 1971.
6. R. Bartlett and M. Garzon, *Monomial cellular automata*, Complex Systems **7** (1993), no. 5, 367–388.
7. ———, *Bilinear cellular automata*, Complex Systems **9** (1995), no. 6, 455–476.
8. C. H. Bennett, *Logical reversibility of computation*, IBM Journal of Research and Development **17** (1973), no. 6, 525–532.
9. E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning ways for your mathematical plays. Vol. 2*, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982, Games in particular.
10. A. W. Burks, *Programming and the theory of automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 65–83 (Essay Two).
11. ———, *Towards a theory of automata based on more realistic primitive elements*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 84–102 (Essay Three).
12. ———, *Von neumann's self-reproducing automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 3–64 (Essay One).
13. J. Cocke and M. Minsky, *Universality of tag systems with p=2*, J. ACM **11** (1964), no. 1, 15–20.
14. E. F. Codd, *Cellular automata*, Academic Press, New York, 1968.
15. M. Cook, *Universality in elementary cellular automata*, Complex Systems **15** (2004), 1–40.
16. M. D. Davis, *A note on universal turing machines*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 167–175.
17. ———, *The universal computer: The road from leibniz to turing*, W. W. Norton & Company, 2000.
18. M. Delorme, *An introduction to cellular automata: some basic definitions and concepts*, Cellular automata (Saissac, 1996) (M. Delorme and J. Mazoyer, eds.), Kluwer Acad. Publ., Dordrecht, 1999, pp. 5–49.
19. J.C. Delvenne, P. Kurka, and V. D. Blondel, *Decidability and universality in symbolic dynamical systems*, Fundam. Inform. **74** (2006), no. 4, 463–490.
20. A. K. Dewdney, *The cellular automata programs that create wireworld, rugworld and other diversions*, Scientific American **262** (1990), 146–149.
21. Jean-Christophe Dubacq, *How to simulate turing machines by invertible one-dimensional cellular automata*, Int. J. Found. Comput. Sci. **6** (1995), no. 4, 395–402.
22. B. Durand and Zs. Róka, *The game of life: universality revisited*, Cellular automata (Saissac, 1996) (M. Delorme and J. Mazoyer, eds.), Kluwer Acad. Publ., Dordrecht, 1999, pp. 51–74.
23. J. Durand-Lose, *Reversible cellular automaton able to simulate any other reversible one using partitioning automata*, LATIN '95 (R. A. Baeza-Yates, E. Goles Ch., and P. V. Poblete, eds.), Lecture Notes in Computer Science, vol. 911, Springer, 1995, pp. 230–244.
24. ———, *Automates cellulaires, automates partitions et tas de sable*, Ph.D. thesis, Université Bordeaux I, 1996.
25. ———, *Intrinsic universality of a 1-dimensional reversible cellular automaton*, STACS 97 (Lübeck), Lecture Notes in Comput. Sci., vol. 1200, Springer, Berlin, 1997, pp. 439–450.
26. Patrick C. Fischer, *On formalisms for turing machines*, J. ACM **12** (1965), no. 4, 570–580.
27. E. Fredkin and T. Toffoli, *Conservative logic*, International Journal of Theoretical Physics **21** (1982), 219–253.
28. A. Gajardo and E. Goles Ch., *Universal cellular automaton over a hexagonal tiling with 3 states*, IJAC **11** (2001), no. 3, 335–354.
29. M. Gardner, *Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'*, Scientific American **223** (1970), no. 4, 120–123.

30. K. Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, Monatschefte für Mathematik und Physik **38** (1931), 173–198, reprinted in S. Feferman, W. Dawson, S.C. Kleene, G. Moore, R.M. Solovay, J. van Heijendort (eds.), *Kurt Gödel: Collected Works*, Oxford University Press, Oxford, **1** (1986), 144–195.
31. G. A. Hedlund, *Endormorphisms and automorphisms of the shift dynamical system*, Mathematical Systems Theory **3** (1969), 320–375.
32. Hertling, *Embedding cellular automata into reversible ones*, Unconventional Models of Computation, Springer, 1998 (C. S. Calude, J. Casti, and M. J. Dinneen, eds.), 1998.
33. K. Imai and K. Morita, *A computation-universal two-dimensional 8-state triangular reversible cellular automaton*, Theoretical Computer Science **231** (2000), no. 2, 181–191.
34. J. Kari, *Reversibility of 2D cellular automata is undecidable*, Physica D. Nonlinear Phenomena **45** (1990), no. 1-3, 379–385, Cellular automata: theory and experiment (Los Alamos, NM, 1989).
35. ———, *Reversibility and surjectivity problems of cellular automata*, J. Comput. Syst. Sci. **48** (1994), no. 1, 149–182.
36. ———, *Theory of cellular automata: A survey*, Theoretical Computer Science **334** (2005), 3–33.
37. S. C. Kleene, *Representation of events in nerve nets and finite automata*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 3–41.
38. G. Lafitte, *Gödel incompleteness revisited*, personal communication *(submitted to JAC 2008)*, 2008.
39. C.G. Langton, *Self-reproduction in cellular automata*, Physica D **10** (1984), no. 1-2, 135–144.
40. Y. Lecerf, *Machines de turing réversibles*, C. R. Acad. Sci. Paris **257** (1963), 2597–2600.
41. K. Lindgren and M. G. Nordahl, *Universal computation in simple one-dimensional cellular automata*, Complex Systems **4** (1990), no. 3, 299–318.
42. N. Margolus, *Physics-like models of computation*, Physica D **10** (1984), 81–95.
43. B. Martin, *Construction modulaire d'automates cellulaires*, Ph.D. thesis, École Normale Supérieure de Lyon, 1993.
44. B. Martin, *A universal cellular automaton in quasilinear time and its S-m-n form*, Theoretical Computer Science **123** (1994), no. 2, 199–237.
45. J. Mazoyer, *Computations on one dimensional cellular automata*, Ann. Math. Artif. Intell. **16** (1996), 285–309.
46. ———, *Computations on grids*, Cellular automata (Saissac, 1996), Kluwer Acad. Publ., Dordrecht, 1999, pp. 119–149.
47. J. Mazoyer and Ivan Rapaport, *Inducing an order on cellular automata by a grouping operation*, Discrete Applied Mathematics **91** (1999), no. 1-3, 177–196.
48. J. Mazoyer and Véronique Terrier, *Signals in one-dimensional cellular automata*, Theoretical Computer Science **217** (1999), no. 1, 53–80, Cellular automata (Milan, 1996).
49. W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics **5** (1943), 115–133.
50. D. B. Miller and E. Fredkin, *Two-state, reversible, universal cellular automata in three dimensions*, Conf. Computing Frontiers (N. Bagherzadeh, M. Valero, and A. Ramírez, eds.), ACM, 2005, pp. 45–51.
51. M. Minsky, *Computation: Finite and infinite machines*, Prentice Hall, Englewoods Cliffs, 1967.
52. E. F. Moore, *Machine models of self-reproduction*, Proceedings of Symposia in Applied Mathematics, vol. 14, American Mathematical Society, 1962, pp. 17–33.
53. ———, *Machine models of self-reproduction*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 187–203 (Essay Six).
54. A. Moreira, *Universality and decidability of number-conserving cellular automata*, Theoretical Computer Science **292** (2003), no. 3, 711–721.
55. K. Morita, *A simple construction method of a reversible finite automaton out of fredkin gates, and its related problem*, IEICE Trans. Inf. & Syst. **E73** (1990), no. 6, 978–984.
56. ———, *Reversible simulation of one-dimensional irreversible cellular automata*, Theoretical Computer Science **148** (1995), no. 1, 157–163.
57. K. Morita and M. Harao, *Computation universality of one-dimensional reversible (injective) cellular automata*, IEICE Trans. Inf. & Syst. **E72** (1989), 758–762.
58. K. Morita and K. Imai, *Self-reproduction in a reversible cellular space*, Theoretical Computer Science **168** (1996), no. 2, 337–366.

59. ———, *Number-conserving reversible cellular automata and their computation-universality*, Theoretical Informatics and Applications **35** (2001), no. 3, 239–258.
60. T. Neary and D. Woods, *P-completeness of cellular automaton rule 110*, Proceedings of ICALP 2006, Lecture Notes in Computer Science, vol. 4051, Springer, Berlin, 2006, pp. 132–143.
61. F. Noural and R.S. Kashef, *A universal four-state cellular computer*, IEEE Transactions on Computers **24** (1975), no. 8, 766–776.
62. N. Ollinger, *Two-states bilinear intrinsically universal cellular automata*, Fundamentals of computation theory (Riga, Latvia, 2001) (Berlin) (R. Freivalds, ed.), Lecture Notes in Computer Science, vol. 2138, Springer, 2001, pp. 396–399.
63. ———, *Automates cellulaires : structures*, Ph.D. thesis, École Normale Supérieure de Lyon, 2002.
64. ———, *The quest for small universal cellular automata*, International Colloquium on Automata, languages and programming (Málaga, Spain, 2002) (Berlin) (P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, eds.), Lecture Notes in Computer Science, vol. 2380, Springer, 2002, pp. 318–329.
65. ———, *The intrinsic universality problem of one-dimensional cellular automata*, Symposium on Theoretical Aspects of Computer Science (Berlin, Germany, 2003), Lecture Notes in Computer Science, Springer, Berlin, 2003, (to appear).
66. D. Perrin, *Les débuts de la théorie des automates*, Technique et Science Informatique **14** (1995), 409–433.
67. E. Post, *The two-valued iterative systems of mathematical logic*, Princeton University Press, Princeton, 1941.
68. ———, *Formal reductions of the general combinatorial decision problem*, American Journal of Mathematics **65** (1943), no. 2, 197–215.
69. I. Rapaport, *Inducing an order on cellular automata by a grouping operation*, Ph.D. thesis, École Normale Supérieure de Lyon, 1998.
70. G. Richard, *From turing machine to rule 110: embedding computational power in small cellular automata through particles and collisions*, personal communication *(submitted to JAC 2008)*, 2008.
71. ———, *A particular universal cellular automaton*, personal communication, 2008.
72. D. Richardson, *Tessellations with local transformations*, Journal of Computer and System Sciences **6** (1972), 373–388.
73. C. E. Shannon, *A universal turing machine with two internal states*, Automata Studies (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton, 1956, pp. 157–165.
74. A. R. Smith, III, *Simple computation-universal cellular spaces*, Journal of the ACM **18** (1971), 339–353.
75. K. Sutner, *Universality and cellular automata*, MCU 2004 (M. Margenstern, ed.), Lecture Notes in Computer Science, vol. 3354, Springer, 2004, pp. 50–59.
76. J. W. Thatcher, *Self-describing turing machines and self-reproducing cellular automata*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 103–131 (Essay Four).
77. ———, *Universality in the von neumann cellular model*, Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Urbana, 1970, pp. 132–186 (Essay Five).
78. G. Theyssier, *Automates cellulaires : un modèle de complexités*, Ph.D. thesis, École Normale Supérieure de Lyon, 2005.
79. ———, *How common can be universality for cellular automata?*, STACS 2005, Lecture Notes in Computer Science, vol. 3404, Springer, 2005, pp. 121–132.
80. Tommaso Toffoli, *Computation and construction universality of reversible cellular automata*, J. Comput. Syst. Sci. **15** (1977), no. 2, 213–231.
81. A. M. Turing, *On computable numbers with an application to the entscheidungs problem*, Proceedings of the London Mathematical Society 2 **42** (1936), 230–265.
82. J. von Neumann, *Theory of self-reproducing automata*, University of Illinois Press, Urbana, Ill., 1966, (A. W. Burks, ed.).
83. S. Wolfram, *Universality and complexity in cellular automata*, Physica D. Nonlinear Phenomena **10** (1984), no. 1-2, 1–35, Cellular automata (Los Alamos, N.M., 1983).
84. S. Wolfram, *A new kind of science*, Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.
85. D. Woods and T. Neary, *On the time complexity of 2-tag systems and small universal turing machines*, FOCS 2006, IEEE Computer Society, 2006, pp. 439–448.
86. ———, *The complexity of small universal turing machines*, Computability in Europe (S. B. Cooper, B. Löwe, and A. Sorbi, eds.), Lecture Notes in Computer Science, vol. 4497, Springer, 2007, pp. 791–799.

A.4 Collisions and their Catenations : Ultimately Periodic Tilings of the Plane

Version rapporteur de [C7], un article réalisé en collaboration avec G. Richard consacré à l'étude des fonds, particules et collisions dans les automates cellulaires vus comme des pavages réguliers du plan, à leur combinatoire et à ses applications algorithmiques. Ces résultats ont été présentés à IFIP-TCS'2008.

Collisions and their Catenations: Ultimately Periodic Tilings of the Plane

Nicolas Ollinger and Gaétan Richard

Laboratoire d'informatique fondamentale de Marseille (LIF),
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France
{nicolas.ollinger,gaetan.richard}@lif.univ-mrs.fr

Abstract. Motivated by the study of cellular automata algorithmic and dynamics, we investigate an extension of ultimately periodic words to two-dimensional infinite words: collisions. A natural composition operation on tilings leads to a catenation operation on collisions. By existence of aperiodic tile sets, ultimately periodic tilings of the plane cannot generate all possible tilings but exhibit some useful properties of their one-dimensional counterparts: ultimately periodic tilings are recursive, very regular, and tiling constraints are easy to preserve by catenation. We show that, for a given catenation scheme of finitely many collisions, the generated set of collisions is semi-linear.

1 Introduction

The theory of regular languages, sets of one-dimensional sequences of letters sharing some regularities, has been well studied since the fifties. Finite state machines [18], regular languages [14, 5], computing devices with bounded memory, monadic second-order logic [4]: various point of views lead to a same robust notion of regular languages. The concept extends to infinite words and various other one-dimensional structures. Unfortunately, when considering two-dimensional words – partial mappings from the plane $\mathbb{Z}^2$ to a finite alphabet – such a robust common object fails to emerge: automata on the plane, picture languages, second-order logic, all lead to different notions of regular languages [9]. A first difficulty arises from the definition of a finite word: should it be any partial mapping with a finite support? Should it be rectangles filled with letters? Should it be any mapping with a connected support for some particular connexity notion? A second difficulty arises from the complexity of two-dimensional patterns: in the simplest case of uniform local constraints, *i.e. tilings*, knowing whether a given finite pattern is a factor of a valid tiling (of the whole plane) is already undecidable [1].

In the present paper, we investigate a particular family of recursive tilings of the plane endowed with a catenation operation. Our definition of an ultimately periodic tiling, a *collision*, is inspired by geometrical considerations on one-dimensional cellular automata space-time diagrams and tilings. It can be

thought of as an extension of the notion of ultimately periodic bi-infinite words to two-dimensional words. These objects provide a convenient tool to describe synchronization problems in cellular automata algorithmic.

One-dimensional cellular automata [13] are dynamical systems whose configurations consist of bi-infinite words on a given finite alphabet. The system evolves by applying uniformly and synchronously a locally defined transition rule. The value at each position, or *cell*, of a configuration only depends on the values of the cells on its neighborhood at the previous time step. To discuss the dynamics or to describe algorithmic constructions, it is often convenient to consider space-time diagrams rather than configurations. A space-time diagram is a drawing of a particular orbit of the system: configurations are depicted one on top of the other, from bottom to top, by successively applying the transition rule, as depicted on Fig. 1. This representation permits to draw away the time-line and discuss the structure of emerging two-dimensional patterns. Formally, this is equivalent to consider tilings of half the plane with a special kind of local constraint, oriented by the time-line.

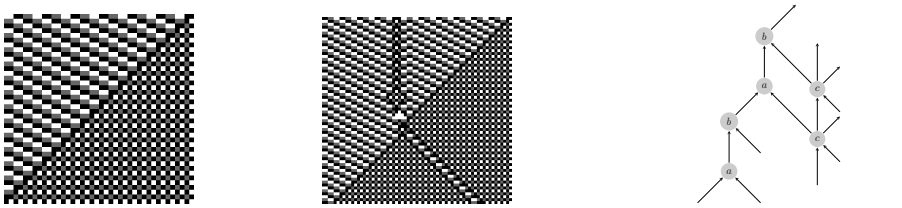
Time goes from bottom to top. Each letter is represented by a different color.

Fig. 1. Space-time diagram of a one-dimensional cellular automaton

Let us give first an informal overview of what collisions are and where they come from. An ultimately periodic configuration consists of two infinite periodic words separated by a finite non-periodic word. As transitions of cellular automata are locally defined, the image of an ultimately periodic configuration is an ultimately periodic configuration such that: for each periodic part, the period in the image divides the period in the preimage; for the non-periodic part, it can only grow by a finite size depending on the local rule. If, by iterating the transition rule of the cellular automaton, the size of the non-periodic part of the configurations remains bounded, then the orbit of the ultimately periodic configuration is, up to a translation, ultimately periodic. When considering this ultimately periodic behavior from the space-time diagram point of view, one can see some kind of *particle*: a localized structure moving with a rational slope in a periodic *background* environment, as depicted on Fig. 2a.

As particles are ultimately periodic configurations, one can construct more complicated configurations by putting particles side by side, ensuring that the non-periodic parts are far enough from each other, and that the periodic parts of two particles put side by side are the same and well aligned. If the non-periodic part of several particles (two or more) becomes near enough in the orbit, complex interactions might occur. If the interaction is localized in both space and time, as depicted on Fig. 2b, this interaction is called a *collision*.

Particles and collisions provide a convenient tool in the study of cellular automata. When constructing two-dimensional cellular automata, like in historical



(a) particle (b) collision (c) network of collisions

Fig. 2. Particles and collisions generated by ultimately periodic configurations

constructions of von Neumann [20] and Codd [6], particles are a convenient way to convey quanta of information from place to place. The most well known example of particle is certainly the glider of the Game of Life used by Conway et al. to embed computation inside the Game of Life [2] by using particular behavior of glider collisions. When using one-dimensional cellular automata to recognize languages or to compute functions, a classical tool is the notion of signal introduced by Fischer [8] and later developed by Mazoyer and Terrier [16, 17]: signals and their interactions are simple kinds of particles and collisions. Particles appear even in the classification of cellular automata dynamics: in its classification [21], Wolfram identifies what he calls class 4 cellular automata where *"(...) localized structures are produced which on their own are fairly simple, but these structures move around and interact with each other in very complicated ways. (...)"* A first study of particles interaction was proposed by Boccara et al. [3], latter followed by Crutchfield et al. [12]: these works focus on particles and bounding the number of possible collisions they can produce. Finally, the proof by Cook of the universality of rule 110 [7] is a typical construction involving a huge number of particles and collisions: once the gadgets and the simulation are described, the main part of the proof consists of proving that particles are well synchronized and that collisions occur exactly as described in the simulation.

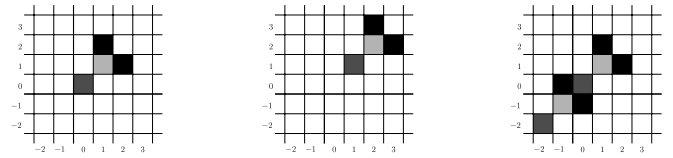
When dealing with space-time diagrams consisting of only particles and collisions, a second object is often used: a planar map describing the collisions and their interactions. When identifying particles and collisions in space-time diagrams, in the style of Boccara et al. [3], one builds the planar map to give a compact description of the diagram, as depicted on Fig. 2(c). When describing algorithmic computation, in the style of Fischer [8], one describes a family of planar maps as a scheme of the produced space-time diagrams.

The aim of the present paper is to define particles and collisions, describe how collisions can be catenated, introduce collisions schemes as planar maps and discuss the construction of finite catenations from collisions schemes. All the necessary material is defined in section 2 followed by basic catenation of tilings in section 3. Collisions and their catenations are formally introduced in section 4. The main result on catenation is presented in section 5.

2 Definitions

In the remaining of this paper, every discussion occurs in the *two-dimensional plane* $\mathbb{Z}^2$ partially colored with the letters of a given finite alphabet Σ. A *pattern* is a subset of $\mathbb{Z}^2$. A *cell* c of a given pattern P is an element $c \in P$. A *vector* is an element of the group $(\mathbb{Z}^2, +)$ of translations in the plane. A *coloring* $\mathcal{C}$ is a partial map from $\mathbb{Z}^2$ to Σ. The *support* of a *coloring* $\mathcal{C}$ is denoted by $\mathrm{Sup}(\mathcal{C})$, its restriction to a pattern P is denoted by $\mathcal{C}_{|P}$.

The *translation* $u \cdot \mathcal{C}$ of a coloring $\mathcal{C}$ by a vector u is the coloring with support $\mathrm{Sup}(\mathcal{C}) + u$ such that, for all $z \in \mathrm{Sup}(\mathcal{C})$, it holds $(u \cdot \mathcal{C})(z + u) = \mathcal{C}(z)$. The *disjoint union* $\mathcal{C} \oplus \mathcal{C}'$ of two colorings $\mathcal{C}$ and $\mathcal{C}'$ is the coloring with support $\mathrm{Sup}(\mathcal{C}) \cup \mathrm{Sup}(\mathcal{C}')$ such that, for all $z \in \mathrm{Sup}(\mathcal{C})$, it holds $\mathcal{C} \oplus \mathcal{C}'(z) = \mathcal{C}(z)$ and for all $z \in \mathrm{Sup}(\mathcal{C}')$, it holds $\mathcal{C} \oplus \mathcal{C}'(z) = \mathcal{C}'(z)$. Colorings and their operations are depicted on Fig. 3.



(a) a coloring $\mathcal{C}$ (b) $\binom{1}{1} \cdot \mathcal{C}$ (c) $\mathcal{C} \oplus \binom{-2}{-2} \cdot \mathcal{C}$

Fig. 3. Colorings, translations and disjoint unions

A *tiling constraint* is a pair (V, Υ) where V is a finite pattern and Υ is a subset of Σ^V. A coloring $\mathcal{C}$ *satisfies* a tiling constraint (V, Υ) if for each vector $u \in \mathbb{Z}^2$ such that V is a subset of $\mathrm{Sup}(u \cdot \mathcal{C})$, it holds $(u \cdot \mathcal{C})_{|V} \in \Upsilon$. For now on we fix a tiling constraint (V, Υ). A *tiling* is a coloring with support $\mathbb{Z}^2$ that satisfies the tiling constraint. For any pattern P, the *neighborhood* along the constraint (V, Υ) is defined as $\partial P = P \cup \{p + v | p \in P \text{ and } v \in V\}$.

In the following, for geometrical considerations, we will implicitly use variations of discrete forms of the Jordan curve theorem [15]. Two points $\binom{x}{y}, \binom{x'}{y'} \in \mathbb{Z}^2$ are *4-connected* if $\binom{|x-x'|}{|y-y'|} \in \left\{\binom{1}{0}, \binom{0}{1}\right\}$, *8-connected* if $\binom{|x-x'|}{|y-y'|} \in \left\{\binom{1}{0}, \binom{0}{1}, \binom{1}{1}\right\}$. A pattern P is *4-connected*, *resp. 8-connected*, if for each pair of points $z, z' \in P$, there exists a 4-connected, *resp.* 8-connected, path of points of P from z to z'. The discrete Jordan curve theorem states that any non empty 4-connected closed path separates the plane into two 8-connected patterns, the interior and exterior of the path. More generally, a *frontier* is a 4-connected pattern separating the plane into n 8-connected patterns, its *borders*.

3 Catenation of tilings

Let (V, Υ) be a tiling constraint and C a set of colorings satisfying this constraint. To generate tilings by catenating colorings in C, the idea is to construct a patchwork of colorings by cutting portions of coloring and glue them together so that tiling constraints are preserved. A simple patchwork of 2 tilings is depicted on Fig. 4.

(a) coloring ■ (b) coloring ■ (c) blueprint (d) patchwork

Fig. 4. A patchwork

Definition 1. *A* patchwork *is a tiling $\mathcal{T}_\phi$ defined for each $z \in \mathbb{Z}^2$ by $\mathcal{T}_\phi(z) = \phi(z)(z)$ where $\phi : \mathbb{Z}^2 \to C$ is the* blueprint *of the patchwork such that:*

1. $\forall \mathcal{C} \in C, \quad \partial\phi^{-1}(\mathcal{C}) \subseteq \mathrm{Sup}(\mathcal{C})$;
2. $\forall z \in \mathbb{Z}^2, \forall v \in V, \quad \phi(z)(z+v) = \phi(z+v)(z+v)$.

Patchworks provide a convenient way to combinatorially generate tilings from a set of valid colorings without knowing explicitly the tiling constraint: it is sufficient to know a super-set of the tiling neighborhood V and to cut colorings on a big enough boundary containing the same letters.

Topology is a classical tool of symbolic dynamics [11], tilings being exactly the shifts of finite type for two-dimensional words. The set of colorings is endowed with the so called Cantor topology: the product of the discrete topology on $\Sigma \cup \{\perp\}$ where $\perp$ denotes undefined color. This topology is compatible with the following distance on colorings: $d(\mathcal{C}, \mathcal{C}') = 2^{-\min\{|z|, \mathcal{C}(z) \neq \mathcal{C}'(z)\}}$. Let $\mathcal{O}_\mathcal{C}$ be the set of colorings $\mathcal{C}'$ such that $\mathcal{C}'_{|\mathrm{Sup}(\mathcal{C})} = \mathcal{C}_{|\mathrm{Sup}(\mathcal{C})}$. The set of $\mathcal{O}_\mathcal{C}$ for colorings $\mathcal{C}$ with a finite support is a base of clopen sets for the given compact perfect topology.

Proposition 1. *The set of patchworks over C is a compact set. Furthermore, it contains the tilings of the closure of C.*

Proof. Let $\mathcal{T}_i$ be a sequence of patchworks over C converging to a limit tiling $\mathcal{T}$. Consider the blueprints ϕ_i of these patchworks. For each cell $z \in \mathbb{Z}^2$, let v_z be the element $(-z \cdot \mathcal{T})_{|V}$ of Υ. Let $\phi(z)$ be any $\phi_i(z)$ such that $(-z \cdot \phi_i(z))_{|V} = v_z$ – such a $\phi_i(z)$ always exists by definition of patchworks as $\mathcal{T}_i$ converges to $\mathcal{T}$. The map ϕ is a blueprint for $\mathcal{T}$.

Let $\mathcal{C}_i$ be a sequence of colorings in C converging to a limit tiling $\mathcal{T}$. For each $\mathcal{C}_i$, let P_i be the largest pattern, for inclusion, such that $\mathcal{C}_{i|P_i} = \mathcal{T}_{|P_i}$. As the sequence $\mathcal{C}_i$ converges to $\mathcal{T}$, the sequence P_i converges to $\mathbb{Z}^2$. Without

loss of generality, consider that P_i is an increasing sequence of patterns. For each i let $\delta(i)$ be the smallest j such that $\partial P_i \subseteq P_j$. Consider $P'_n = P_{\delta^n(1)}$, an increasing sub-sequence of P_i. Construct a blueprint ϕ as follows: for all $z \in \mathbb{Z}^2$, let $\phi(z) = P'_{\min\{n|z\in P'_n\}}$. By construction, this blueprint is valid and its patchwork is $\mathcal{T}$. ■

Corollary 1. *Let $\mathcal{O}_i$ be a base of open sets of colorings and C be a set of colorings containing at least one element of each $\mathcal{O}_i$. The set of patchworks over C is the whole set of tilings.* ■

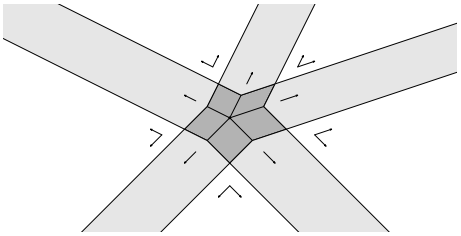
In particular, the set of tiling constraints Υ, viewed as colorings, generates the whole set of tilings. The larger set of colorings with finite support generates the whole set of tilings. But this approach is heterogeneous: we combine colorings to obtain tilings. Can we restrict ourselves to combinations of tilings? More precisely, given a tiling constraint, can we recursively construct a recursive family of tilings T such that the set of patchworks over T is the whole family of tilings?

In the case of one-dimensional tilings, replacing $\mathbb{Z}^2$ by $\mathbb{Z}$, it is straightforward that the set of ultimately periodic tilings generates the whole set of tilings: the set of ultimately periodic tilings is a dense set – from any tiling $\mathcal{T}$ and any finite pattern P, one can construct an ultimately periodic tiling $\mathcal{T}'$ such that $\mathcal{T}_{|P} = \mathcal{T}'_{|P}$. In the case of two-dimensional tilings, due to the undecidability of the tiling problem [1, 19], there exists no such family. This result prohibits us to obtain a recursive set of tilings whose closure under catenation give us the whole set of tilings. Therefore, in the rest of the paper, we search for simplicity rather than being exhaustive.

4 Ultimately periodic tilings

Bi-periodic tilings are among the most regular ones and correspond to the idea of a *background* for cellular automata: a tiling $\mathfrak{B}$ with two non-co-linear periodicity vectors u and v such that $\mathfrak{B} = u \cdot \mathfrak{B} = v \cdot \mathfrak{B}$. As backgrounds are objects of dimension 2, if one wants to mix several backgrounds in a same tiling, the interface between two background is of dimension 1. The most regular kind of interface corresponds to the idea of a *particle*: a tiling $\mathfrak{P}$ with two non-co-linear vectors, the period u of the particle such that $\mathfrak{P} = u \cdot \mathfrak{P}$ and the period v of its backgrounds such that for all position $z \in \mathbb{Z}^2$, the extracted one-dimensional word $(\mathfrak{P}(z + vi))_{i\in\mathbb{Z}}$ is ultimately periodic. Of course, several particles might meet on the plane, leading to objects of dimension 0 that correspond to the idea of a *collision*. In this paper, an ultimately periodic tiling of the plane is such a collision.

Let $\lhd_v(u, u')$ denote the angular portion of the plane, on the right hand side of u, starting in position $v \in \mathbb{Z}^2$ and delimited by the vectors $u, u' \in \mathbb{Z}^2$. Formally, one might geometrically define a collision as follows (and depicted on Fig. 5):



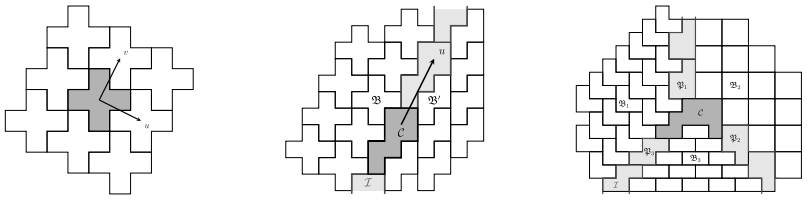
$k = 2$
$u_0 = \binom{1}{2}$
$u_1 = \binom{3}{1}$
$u_2 = \binom{2}{-2}$
$u_3 = \binom{-2}{-2}$
$u_4 = \binom{-2}{1}$

Fig. 5. Defining collisions through vectors

Definition 2. *A* collision *is a tiling* $\mathfrak{C}$ *for which there exists an integer* k *and a finite cyclic sequence of* n *vectors* $(u_i) \in \left(\mathbb{Z}^2\right)^{\mathbb{Z}_n}$ *such that, for all* $i \in \mathbb{Z}_n$, $\mathfrak{C}$ *is* u_i*-periodic in* z*, i.e.* $\mathfrak{C}(z) = \mathfrak{C}(z+u_i)$*, for all positions* z *inside* $\lhd_{ku_i}(u_{i-1}, u_{i+1})$.

Although it corresponds to intuition, this definition made it difficult to effectively use collisions in constructions since it does not identify components of the collision. To overcome this problem, we introduce *constructive* versions of collisions. Ideas behind such definitions is that all elements can be represented with a finite description. A background is entirely determined by two non-collinear vectors of periodicity u and v and by a coloring of finite support $\mathcal{C}$ that tiles the plane along u and v (i.e.$\bigoplus_{i,j\in\mathbb{Z}^2}(iu + jv) \cdot \mathcal{C}$ is a tiling) (see Fig. 6). Such a triple $(\mathcal{C}, u, v)$ is called *background representation.*

The same way, in a particle, the uni-periodic part can be characterised by a vector u and a coloring with finite support $\mathcal{C}$ which repeats along u ($\mathcal{I} = \bigoplus_{k\in\mathbb{Z}} ku\cdot\mathcal{C}$) is a frontier with two borders (L and R). The rest of particle can be described using two backgrounds $\mathfrak{B}$ and $\mathfrak{B}'$. The resulting coloring $\mathfrak{P} = \mathfrak{B}_{|L} \oplus \mathcal{I} \oplus \mathfrak{B}'_{|R}$ is require to be a tiling. Furthermore, we require to have a condition ensuring that the different portion have some common "safety zone". This is done by adding the constraint that the function: $\phi : z \to \begin{cases} \mathfrak{P} & \text{if } z \in \mathrm{Sup}(I) \\ \mathfrak{B} & \text{if } z \in L \\ \mathfrak{B}' & \text{if } z \in R \end{cases}$ is the blueprint of a patchwork. Such a tuple $(\mathfrak{B}, \mathcal{C}, u, \mathfrak{B}')$ is called *particle representation.*



(a) a background (b) a particle (c) a collision

Fig. 6. Principle of construction

For collisions, the idea is basically the same (see Fig. 6) , the characterisation is based on a coloring with finite support $\mathcal{C}$ for the non-periodic part and a finite list of particles. Each particle defines a half-line starting form the center of the collision. The support of all the particles and the center must form a star and each consecutive pair of particles must have a common background to fill the space between them. Some safety zone is also required as in particle. This is formalised in the following definition:

Definition 3. *A* collision representation *is a pair* $(\mathcal{C}, L)$ *where* $\mathcal{C}$ *is a finite pattern,* L *is a finite sequence of* n *particles* $\mathfrak{P}_i = (\mathfrak{B}_i, \mathcal{C}_i, u_i, \mathfrak{B}'_i)$*, satisfying:*

1. $\forall i \in \mathbb{Z}_n, \quad \mathfrak{B}'_i = \mathfrak{B}_{i+1}$*;*
2. *the support of* $\mathcal{I} = \mathcal{C} \oplus \bigoplus_{i \in \mathbb{Z}_n, k \in \mathbb{N}} ku_i \cdot \mathcal{C}_i$ *is a frontier with* n *borders;*
3. *For all* $i \in \mathbb{Z}_n$*, the support of* $\mathcal{C} \oplus \bigoplus_{k \in \mathbb{N}} (ku_i \cdot \mathcal{C}_i \oplus ku_{i+1} \cdot \mathcal{C}_{i+1})$ *is a frontier with two borders: let* P_i *be the border on the right of* $\mathfrak{P}_i$*;*
4. $\mathfrak{C} = \mathcal{I} \oplus \bigoplus_i \mathfrak{B}_{i|P_i}$ *is a tiling;*
5. *the function* $\phi : z \to \begin{cases} \mathfrak{C} & \text{if } z \in \mathrm{Sup}(\mathcal{C}) \\ \mathfrak{P}_i & \text{if } z \in \mathrm{Sup}(\bigoplus_{k \in \mathbb{N}} ku_i \cdot \mathcal{C}_i) \\ \mathfrak{B}_i & \text{if } z \in P_i \end{cases}$ *is the blueprint of a patchwork.*

The set $\mathrm{Sup}(\mathcal{C})$ is called *perturbation* of the collision and $\mathrm{Sup}(\bigoplus_{k \in \mathbb{N}} ku_i \cdot \mathcal{C}_i)$ are called *perturbation* of the particle P_i.

The constructive definitions of particles, backgrounds and collisions provide us with a finite representation that allows us to recursively manipulate them. Contrary to intuition, representations are not invariant by translation. This seems unavoidable since we want to have means of expressing the relative position between two such representations. In the rest of the paper, we will always assume that background, particles and collisions are given by a representation.

5 Finite catenations

A blueprint of finitely many collisions might produce a tiling which is not a collision, however if the blueprint of the patchwork consists of finitely many 8-connected components, the patchwork is a collision. Using representations of collisions, a more regular family of patchworks can be defined: a catenation induces a patchwork combining collisions by binding pairs of similar particles as depicted on Fig. 2.

To "bind" collisions using particles, we need two identical particles facing each other such that the gap between them correspond to a integer number of particles n. Two particles $\mathfrak{P} = (\mathfrak{B}, \mathcal{C}, u, \mathfrak{B}')$ and $\tilde{\mathfrak{P}} = (\tilde{\mathfrak{B}}, \tilde{\mathcal{C}}, \tilde{u}, \tilde{\mathfrak{B}}')$ form a n-*binding* if $\tilde{u} = -u$ (particles are facing each other), $\tilde{\mathcal{C}} = (n-1)u \cdot \mathcal{C}$ (they have the same finite pattern and gap is n repetitions) , $\tilde{\mathfrak{B}} = (n-1)u \cdot \mathfrak{B}'$, $\tilde{\mathfrak{B}}' = (n-1)u \cdot \mathfrak{B}$ (backgrounds are the same). The set $\bigoplus_{0<i<n-1} iu \cdot \mathcal{C}$ is called the *perturbation* of the n-binding.

Since we want to get rid of positions, we introduce the *potential n-binding*. The idea is that given two collisions and one particle for each collision, the particles $\mathfrak{P}_1$ and $\mathfrak{P}_2$ form a *potential n-binding* if up to a translation z, the two particles form an n-binding (i.e. $\mathfrak{P}_1$ and $z \cdot \mathfrak{P}_2$ form a n-binding). One can remark in case of potential n-binding, the translation vector z is unique.

Now the idea is that we can use potential n-binding to construct patchworks since background is bi-periodic and does not cause heavy harm for checking properties on it. The description needs to have collisions as points and particles as lines. Particles can be half-infinite (if they are not part of potential n-binding) or link two collisions. Since we work in the plane, it is sound to require that the constructed element is planar and that the order of particles is compatible with the collisions. At last, we add a connected condition to avoid problem with free parts of the map. This leads to the following definition:

Definition 4. *A* catenation *is a connected planar map where:*

- *vertices are labeled by collisions;*
- *edges are potentially semi infinite;*
- *edges extremities are labeled by particles;*
- *edges order in a vertex is compatible with the order on particles in the corresponding collision.*
- *finite edges (of extremities $\mathfrak{P}_1$ and $\mathfrak{P}_2$) are labeled with an integer n such that $\mathfrak{P}_1$ and $\mathfrak{P}_2$ form a potential n-binding.*

At this point, we want to transform the catenation into a patchwork. For this, let us first study some necessary conditions. Since we deal with a planar map, it is possible to define faces as elements of the dual of catenation. To transform a catenation into a patchwork, it is necessary that every face can be transformed into a patchwork. Since we have potential n_i-bindings, the translation induced between two consecutive collisions is fixed. Since the sequence of collisions in a face is cyclic, it is sound to require that the sequence of corresponding translation sum up to zero when cycling. This will be the first condition. Now, with this condition, it is possible to assign (up to a global constant) a translation to every collision such that all edges are n_i-bindings. With those objects, the basic idea is to construct a patchwork that corresponds to each collision, particle or n_i-binding on its perturbation. This implies that all perturbations does not enforce contradictions. One easy way to get rid of this risk is just to require that all perturbations are distinct (this will be our second condition). If these conditions are met then we speak of *valid* catenation.

Proposition 2. *It is possible to associate a patchwork (and therefore a space-time diagram) to every valid catenation.*

Proof. To prove this result, we shall give a potential blueprint and show that it satisfies the conditions. First of all, the condition on null translation after a round on every faces induce a unique set of translation (up to a constant)

for every collision in the map since the map is connected. At this point, let us consider the collisions with those translations.

The second condition ensure that perturbations of collisions, bindings and particles are disjoint. Thus it is possible to define a blueprint linking any point of such a perturbation to the corresponding collision, particle or binding. Let us now study the points that are not mapped. Since the map is planar and particle (and also bindings) are isolating, every left point belongs to one unique face. On this face, the associated background with particles or collision or bindings present is unique (bindings ensure that two consecutive collisions are the same and collisions ensure this for consecutive particles and bindings). So we map those points to the corresponding background.

The last point is to show that the constructed blueprint does really satisfy the properties for patchwork. The first condition on definition is trivial since the used valid coloration are tilings. Let us go now to the second and main point.

For this last part, let us study the different cases. For example, if we are in a collision $\mathfrak{C}$ perturbation. If the neighborhood is also in $\mathfrak{C}$ perturbation or in perturbation of binding, particle belonging to $\mathfrak{C}$ or even of background with this property, then the neighborhood is by definition equal to the original one of a collision. the only difficult case is when in the neighborhood, there is a perturbation originated from another element. For example let us suppose this elements is in the perturbation from $\mathfrak{C}'$. In this case, in $\mathfrak{C}$ we have in these points some backgrounds or particles. But since perturbations do not overlap, we are in the border of $\mathfrak{C}'$. As we have requested in our constructive version representation to be patchworks, the border of $\mathfrak{C}'$ does correspond to the value of backgrounds or particles present in $\mathfrak{C}'$. By definition of our catenations, the backgrounds and particles are the same so elements of $\mathfrak{C}'$ are the same of those in $\mathfrak{C}$.
The same arguments do also apply for other cases thus ending the proof. ■

At this point, we have both a set of "simple" tilings (the collisions) and an operation generating new tilings from this set (the valid catenation). Despite being intuitive, catenations require to give explicitly the relative positions of collision via the number of repetitions of particles. Intuitively, we would like to give only the collisions involved and their organisation (as in Fig. 2c). With this approach, it is possible to define an alternative to catenation that does not require the number of repetitions to be given. The resulting element is called catenation scheme. Formally, a *catenation scheme* is a catenation whose label on finite edge where erased. Conversely, to go back from a catenation scheme to a catenation, one need to give every finite edge a label. Such elements of $\mathbb{N}^F$ where F is the set finite edges of the catenation scheme is called *affectation*. Moreover, it is called *valid* affectation if the resulting catenation is valid.

For a given catenation scheme, one natural question is whether it correspond to a tiling. To bring an answer one idea is to search for valid affectation of the scheme. In case of finite catenation scheme, we can achieve a very strong characterisation of this set and even compute it.

Theorem 1. *The set of valid affectation of a finite catenation is a recursive semi-linear set (i.e. a finite union of linear sets).*

Proof. To prove the main theorem, we will show that being a valid affectation of finite catenation scheme can be expressed with a formula in Presburger arithmetic (i.e first order logic on integer with addition and comparison). Since the set of solutions of formula in such arithmetic is a recursive semi-linear set [10] this will conclude the proof. One can note that the construction of the solution is explicit even if the complexity is non-elementary.

In our formula, the number of repetitions of each finite edge will correspond to free variables. let us call them $r_1, \dots, r_n$. Since the conditions for valid catenation are for each face, the global formula F will consists on the conjunction of an elementary formula for each face: $F = \wedge_{f\text{face}} F_f$. For each face, let us look at the two conditions. First one (going back to the same point after a turn around the face) can be easily expressed: the translation induced by a particle i is just r_i times the vector of repetition of the particle u_i (just note that the direction of the particles is chosen in the face) which is a known constant. For the translation induced by collision ϵ_c they are know constant. So the formula is on the form $F_{f,1} = \Sigma_{i\text{particles in the face}} u_i r_i + \Sigma_{c\text{collisions}} \epsilon_c = \binom{0}{0}$. For the second condition (non overlap of perturbation) it can be expressed with the conjunction that any pair of points of different perturbations are distinct. In the case of collision perturbation, it is trivial since there is only a finite (and known) number of perturbation points. For bindings, it is more difficult since the set of points can be expressed with a universal quantifier with the following remark, the set of points in the binding's perturbation correspond to the set of points of the particle perturbation $\mathrm{Sup}(\mathcal{C}_i)$ (a finite number) for every integer n multiple of the vector of repetition u_i which is between 0 and the number of repetition r_i. thus the formula is on the form: $\forall x, 0 < x < r_i \Rightarrow \wedge_{p \in \mathrm{Sup}(\mathcal{C}_i)} p + u_i r_i \neq z$ where z are points for the other considered perturbation. The same applies for free particles (just omit the upper bound in the comparison).

With this, we have show how to construct the Presburger formula which conclude the proof. ■

With this theorem we achieve a very strong framework for cellular automata. After have extracted a set of collisions, one can give the desired finite catenation scheme and automatically check the necessary and sufficient conditions for that scheme to exists. This method would make proves far more understandable and could avoid the need to rely on combinatorial proves to ensure validity of intuition. For now, the main limitation of those results are that only the field of finite catenations are treated. One main goal of future work is to achieve such kind of result for infinite catenation schemes. Due to the infinite nature of such elements, such strong a characterisation is excluded but we hope to have sufficient computable conditions for affectation of a wide range of "regular" infinite catenations.

References

1. R. Berger. The undecidability of the domino problem. *Memoirs American Mathematical Society*, 66, 1966.
2. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays. Vol. 2.* Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982. Games in particular.
3. N. Boccara, J. Nasser, and M. Roger. Particlelike structures and their interactions in spatiotemporal patterns generated by one-dimensional deterministic cellular-automaton rules. *Phys. Rev. A*, 44(2):866–875, 1991.
4. J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science, Berkley, 1960*, pages 1–11. Standford University Press, 1962.
5. N. Chomsky. Three models for the description of language. *Information Theory, IEEE Transactions on*, 2(3):113–124, 1956.
6. E. F. Codd. *Cellular Automata.* Academic Press, New York, 1968.
7. M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.
8. P. C. Fischer. Generation of primes by a one-dimensional real-time iterative array. *Journal of the ACM*, 12(3):388–394, 1965.
9. D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, pages 215–267. Springer-Verlag, Berlin, 1997.
10. S. Ginsburg and E. H. Spanier. Semigroups, presburger formulas, and languages. *Pacific Journal of Mathematics*, 16:285–296, 1966.
11. G. A. Hedlund. Endormorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3:320–375, 1969.
12. W. Hordijk, C. R. Shalizi, and J. P. Crutchfield. Upper bound on the products of particle interactions in cellular automata. *Phys. D*, 154(3-4):240–258, 2001.
13. J. Kari. Theory of cellular automata: a survey. *Theoretical Computer Science*, 334:3–33, 2005.
14. S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.
15. T. Y. Kong and A. Rosenfeld. Digital topology: introduction and survey. *Comput. Vision Graph. Image Process.*, 48(3):357–393, 1989.
16. J. Mazoyer. Computations on cellular automata: some examples. In *Cellular automata (Saissac, 1996)*, pages 77–118. Kluwer Acad. Publ., Dordrecht, 1999.
17. J. Mazoyer and V. Terrier. Signals in one-dimensional cellular automata. *Theoretical Computer Science*, 217(1):53–80, 1999. Cellular automata (Milan, 1996).
18. M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, Englewoods Cliffs, 1967.
19. R. M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12:177–209, 1971.
20. J. von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana, Ill., 1966.
21. S. Wolfram. Universality and complexity in cellular automata. *Physica D. Nonlinear Phenomena*, 10(1-2):1–35, 1984. Cellular automata (Los Alamos, N.M., 1983).

A Appendix for the referee

In this appendix, we provide a full example of use of the framework presented in this paper. To do this, we show how to construct an automaton with a specific behavior (in our example, calculate one step of the Syracuse sequence) and prove that the constructed automaton really works.

A.1 Constructing the automaton

The idea is to describe the desired behavior of our automaton using geometrical signals and then to implement a cellular automaton rule with this behavior. Suppose that we have encoded the current value of the sequence in a gap between two vertical signals (see Fig. 7). Suppose also that the start of the step is given by a full-speed left-going signal coming from left. This signal cross the right boundary in α and arrive in the left one in β. Now, let us distinguish both cases: in the odd case, we send a full speed right going signal from β and a signal with slope $1/2$ from α. Their intersection will mark 3 time the initial distance. In the even case, we send at the same time a $1/3$ slope signal from β and a full speed signal which reflex again the left boundary. Those two signal will intersect at half the initial size. In our sketch, we also add some other signals for cleaning purpose or convenience. Note that our solution is neither unique nor optimal but merely an interesting example.

From the geometric figure, we want to construct an automaton with 1 background, 7 particles and 8 collisions. The rest of the section is devoted to construct a rule exhibiting these objects. The choice of transitions is detailed in Fig 8a for each elements, the set of used transition (that do not map on white) are given along an extract of the object space-time diagram (Fig. 8b).

The chosen background is just uniformly white. For full speed or vertical particles, the chosen implementation is nearly always to take a supplementary state for each particle. One can note that particle c is made of two different altering states. These states allows to remember the parity of the length. For other slopes, the particle is constructed by using several states for alternating staying and advancing at the correct rate. One can note that particles g also contains a bit of information in its repetition (as particle c).

A.2 Catenations

For this automaton, we shall look at the two catenation scheme induced by the geometric signal approach (see Fig. 7). The faces present in these schemes are depicted on Fig. 8d. In this section, we shall give the detailed study of face F_1.

First of all, let us express the fact that we are going back to the same point after a turn around the face. For this face, we have three bindings. The oriented repetition axes of these particles are: $u_a = \binom{0}{-1}$, $u_c = \binom{-2}{2}$ and $u_e = \binom{1}{1}$. Each of the three collision induce a translation between the two present particles.

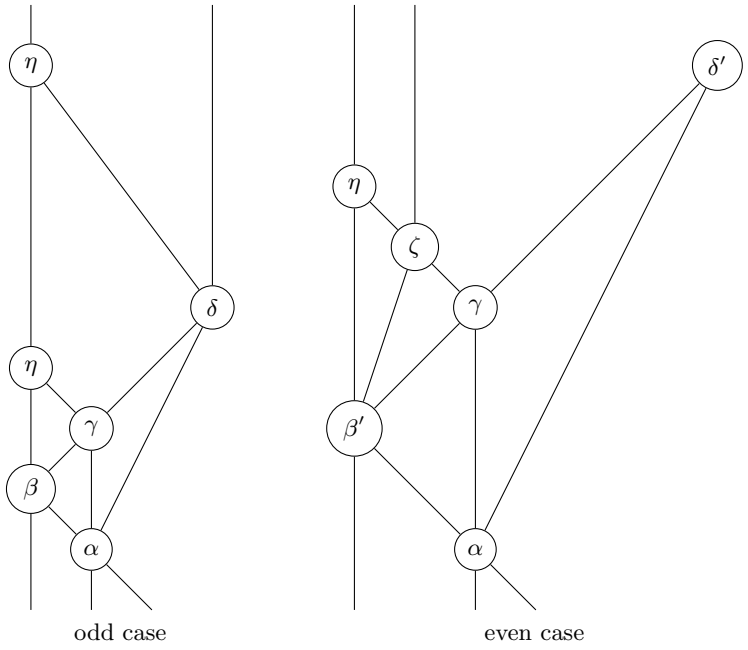


Fig. 7. Computing one step of the Syracuse sequence with geometric signals

Here, we have $\epsilon^{\alpha}_{a,c} = \binom{-2}{-3}$ $\epsilon^{\beta}_{c,e} = \binom{-1}{5}$ and $\epsilon^{\gamma}_{e,a} = \binom{3}{0}$. The resulting equation is obtained by summing up all the contributions and requesting a null sum:

$$F_0 : (r_a - 1)u_a + \epsilon^{\alpha}_{a,c} + (r_c - 1)u_c + \epsilon^{\beta}_{c,e} + (r_e - 1)u_e + \epsilon^{\gamma}_{e,a} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

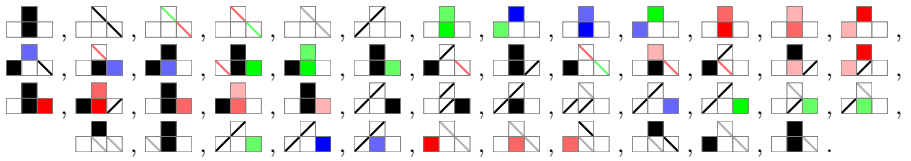
Once this is done, it is possible to associate to each collision a translation t_i which only depends linearly on the number of repetition of particles inside bindings. Then it is time for expressing the fact that perturbations are distinct. The formula are different along the considered elements. Simplest one are describing the non-overlap of collisions perturbations. For example, the equation for perturbations of α and β is

$$F_{\alpha,\beta} : \bigwedge_{x \in \mathrm{Sup}(t_\alpha \cdot \mathcal{C}_\alpha)} \bigwedge_{y \in \mathrm{Sup}(t_\beta \cdot \mathcal{C}_\beta)} x \neq y$$

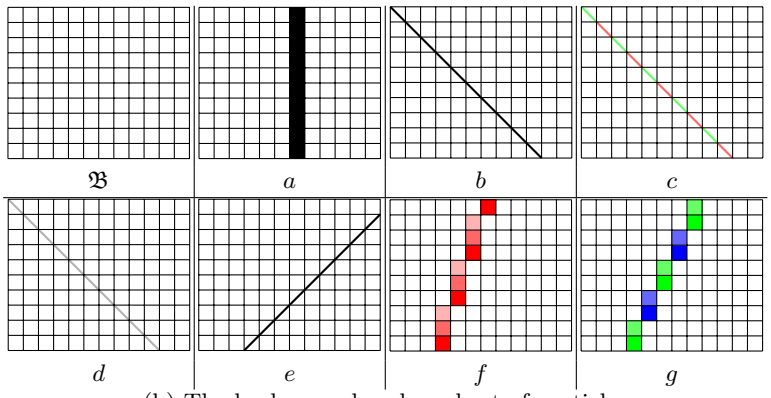
. The same can be done between collisions and bindings supports. For example, the equation between perturbation of α and e is on the form:

$$F_{\alpha,e} : \bigwedge_{x \in \mathrm{Sup}(t_\alpha \cdot \mathcal{C}_\alpha)} \forall p \in \mathbb{N}, 0 \leq p < r_e \Rightarrow \bigwedge_{y \in \mathrm{Sup}(\mathcal{C}_e)} x \neq y + pu_e$$

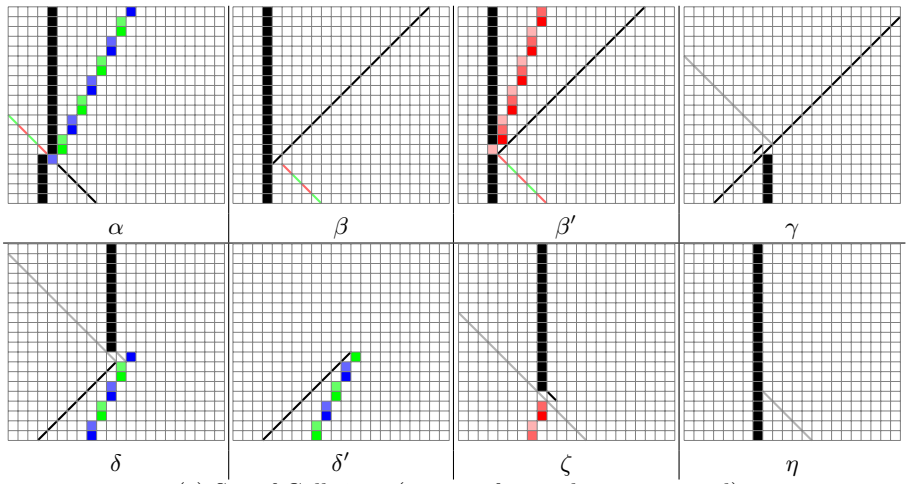
The same formula can be expressed in the case of two bindings.



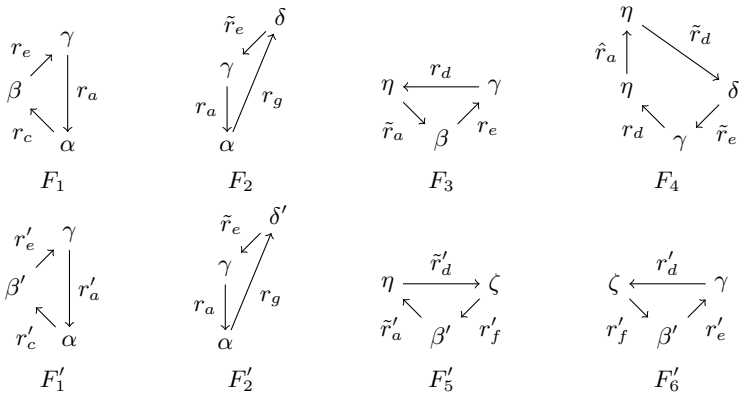
(a) Set of constraints: all non defined behavior maps on □



(b) The background and used set of particles



(c) Set of Collisions (names of particles are omitted)



(d) Faces present in catenation schemes

Fig. 8. Elements of the automaton

The equation for the face is made by doing the conjunction of all the previous formula:

$$F : F_0 \wedge \bigwedge_{x,y\in\{\alpha,\beta,\gamma,a,e,c\},x\neq y} F_{x,y}$$

At this point, one can use quantifier elimination and find the set of values for r_a, r_e and r_c that satisfy the formula. We obtain that the set of solution is on the form:

$$\begin{pmatrix} n_c \\ n_e \\ n_a \end{pmatrix} \in \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

The same can be done with all other faces. The results are given in Fig. 9. In our simple case, all solutions are just linear set and half-infinite faces do not bring any further constraints so they were omitted in this tabular.

$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 6 \\ 2 \\ 0 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 6 \\ 2 \\ 0 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 0 \\ 2 \\ 1 \\ 1 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$
$F_1(r_c, r_e, r_a)$	$F_2(r_a, \tilde{r}_e, r_g)$	$F_3(\tilde{r}_a, r_d, r_e)$	$F_4(r_d, \hat{r}_a, \tilde{r}_d, \tilde{r}_e)$
$\begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 11 \\ 0 \\ 2 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \mathbb{N} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$
$F'_1(r'_c, r'_e, r'_a)$	$F'_2(r'_a, \tilde{r}'_e, r'_g)$	$F'_3(\tilde{r}'_a, \tilde{r}'_d, r'_f)$	$F'_4(r'_f, r'_d, r_e)$

Fig. 9. Validity set for faces

At this point, one want to study the set of validity for the whole scheme. One can note that it correspond to make the synchronized product of the set of validity of the faces. The resulting set found is depicted in Fig. 10. Note that since all repetitions numbers must be positive, the even case has implicit requirement $x > 1$.

A.3 Behavior

The previous result give us an idea of possible behavior and when they occur. To finish our construction and proof, we need to relate the number of particles to the encode value. In the odd case, the fact that $r_c = x$ induce a distance of $2x + 3$ between the two particles a whereas the value $\tilde{r}_d = 6x + 4$ correspond to a distance of $6x + 10$. In the odd case, $r'_c = x$ correspond to an input of $2x + 4$ and $\tilde{r}_d = x - 2$ to a distance of $x + 2$. Thus we have proved that our automaton can compute one step of the Syracuse sequence (for sufficiently large values of input).

The main advantage of this method of construction is that the proof does follow directly the method of construction and does not require to make study

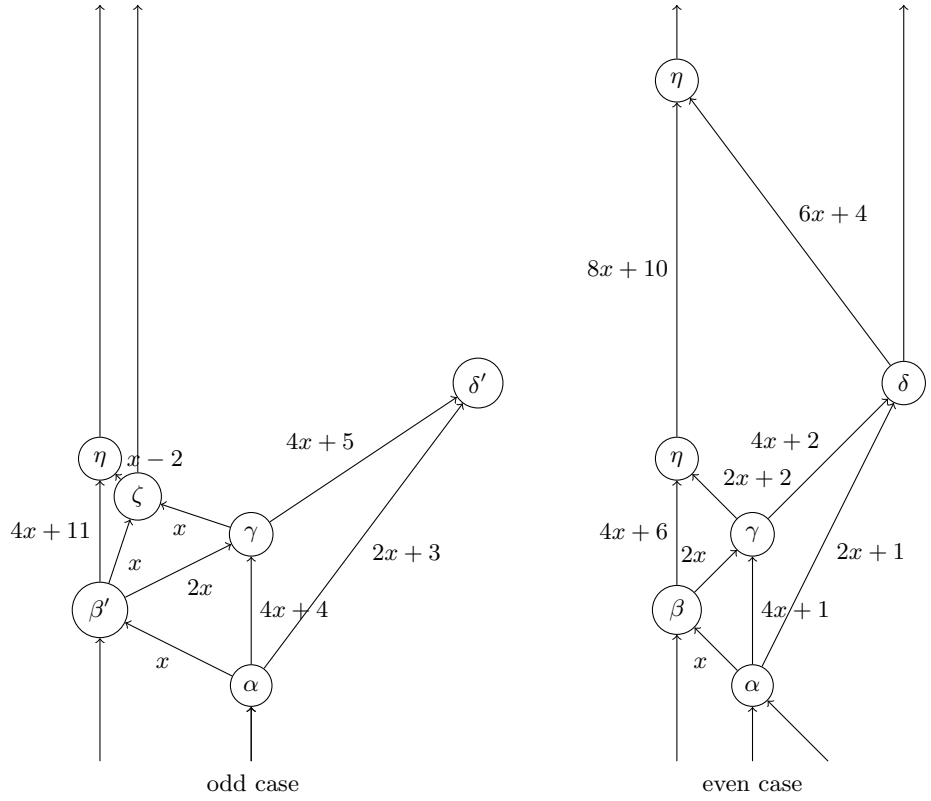


Fig. 10. Solution of the faces

the local details (since they are hidden in the resolution of the validity set). Thus making the proof easier to follow.

A.5 Automata on the Plane vs Particles and Collisions

Version courante de [U4], une note écrite en collaboration avec G. Richard. Dans cette note, nous nous intéressons aux pavages réguliers introduits dans [C7], fonds, particules et collisions, et nous montrons qu'ils sont caractérisés précisément par les coloriages réalisés par les automates finis, les automates à 1 compteur et les automates apériodiques à 2 compteurs.

Automata on the Plane vs Particles and Collisions

N. Ollinger and G. Richard

Laboratoire d'informatique fondamentale de Marseille (LIF),
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France

Abstract

In this note, the coloring of the plane by finite sequential machines is compared to previously introduced notions of ultimately periodic tilings of the plane. Finite automata with no counter characterize exactly biperiodic tilings. Finite automata with one counter characterize exactly particles: periodic colorings that are ultimately periodic in every direction. Finite automata with two counters and aperiodic colorings characterize exactly collisions: ultimately periodic tilings of the plane.

Key words: counter automata, languages, two-dimensional

Introduction

In [1], motivated by the study of space-time diagrams of cellular automata, we introduced collisions as a practical notion of ultimately periodic tiling of the plane: an extension of the notion of ultimately periodic biinfinite words to infinite bidimensional tilings. Intuitively, ultimately periodic words and collisions share the property to be locally almost everywhere periodic in every direction. Imagine that you are walking on the plane, trying to color it according to a collision: you only need to keep a finite information, your position inside the biperiodic pattern, plus a way to store your distance to the boundaries: places where one should switch from a biperiodic region to another. All you need to know about this distance is when it becomes equal to zero. Therefore, counter machine coloring the plane can certainly encode every collision.

In this note, we explore the analogy between regular tilings and colorings by counter machines. A map automaton over a free monoid is a deterministic counter machine that starts in a given initial state with empty counters on the unit element of the monoid. The automaton walks on the monoid by firing

transitions labelled by the generator associated to each of its move. Such an automaton can certainly color each element of the monoid with a finite set of colors according to its state. To color a group, like the euclidian plane $\mathbb{Z}^2$, with a map automaton, we simply choose a monoid presentation for the group and require the automaton to be compatible with the group structure – that is, to have the same state and counter values on two elements of the free monoid corresponding to a same element of the group.

As expected, in the case of biinfinite words, map automata with no counter capture periodic words; map automata with one counter capture ultimately periodic words; and map automata with two counters can paint arbitrarily complex recursive tilings. In the case of bidimensional tilings, map automata coloring the plane with a periodicity vector act like a finite family of map automata on biinfinite words. Thus, map automata with no counter capture biperiodic tilings; map automata with one counter capture particles. In the case of two counters automata, if the coloring is periodic then it can be arbitrarily recursively complex. However, aperiodic map automata with two counters capture collisions. In the case of aperiodic colorings, the two counters of a map automata acts like a compas pointing to the origin cell using finitely many biperiodic quadrants: this is a collision.

The note is organized as follows. In section 1, we introduce map automata on monoid presentations and some of their properties. Section 2 studying map automata on $\mathbb{Z}$ and section 3 studying map automata on $\mathbb{Z}^2$ are constructed symmetrically: notions of regular colorings are first defined before a sequential study of automata with 0, 1 and 2 counters.

1 Definitions

Let Σ be a finite alphabet, Σ^* is the free monoid generated by Σ, the set of words on Σ, with *empty word* ϵ. The catenation of $u \in \Sigma^*$ and $v \in \Sigma^*$ is denoted as uv. A *finite monoid presentation* is a pair (G, R) where G is a finite alphabet of *generators* and $R \subseteq G^* \times G^*$ is a finite set of *relators*. The *monoid* $\mathcal{G} = \langle G|R\rangle$ *associated to the presentation* is the largest monoid satisfying the relators equations, *i.e.* such that for each $(u, v) \in R$, $u = v$ in this monoid.

A $\mathcal{G}$-*coloring* is a mapping $c : \mathcal{G} \to \Sigma$. It is *periodic*, with period $z \in \mathcal{G} \setminus \{\epsilon\}$, if for all $z' \in \mathcal{G}$, $c(zz') = c(z')$. Is is *aperiodic* if it is not periodic. It is *biperiodic*, with periods $z, z' \in \mathcal{G} \setminus \{\epsilon\}$, if z and z' are two non-collinear periods, *i.e.* there does not exist $k, k' \in \mathbb{Z}^+$ such that $z^k = z'^{k'}$.

Let $\Upsilon = \{0, +\}$ and $\Phi = \{-, 0, +\}$ be respectively the set of *test values* and *counter operations*. Let $\mathbf{0}$ denote the constant k-uple $(0, \ldots, 0)$. For all $\phi \in \Phi^k$,

testing τ and *modifying* actions are defined for all $i \in \mathbb{Z}_k$, $v \in \mathbb{N}^k$ and $j \in \mathbb{N}$ as:

$$\tau(j) = \begin{cases} 0 \text{ if } j = 0 \\ + \text{ if } j > 0 \end{cases} \qquad \theta_\phi(v)(i) = \begin{cases} \max(0, v(i) - 1) & \text{if } \phi_i = - \\ v(i) & \text{if } \phi_i = 0 \\ v(i) + 1 & \text{if } \phi_i = + \end{cases}$$

A *k-counter map automaton on the alphabet* Σ is a tuple $(\Sigma, k, S, s_0, \delta)$ where $k \in \mathbb{N}$ is the number of counters, S is a finite *set of states* with *initial state* $s_0 \in S$ and $\delta : S \times \Upsilon^k \times \Sigma \to S \times \Phi^k$ is the *transition rule* of the automaton. Its *transition function* $f : S \times \mathbb{N}^k \times \Sigma^* \to S \times \mathbb{N}^k$ is recursively defined on Σ^* by $f(s, v, \epsilon) = (s, v)$ and $f(s, v, za) = (s'', \theta_\phi(v'))$ where $f(s, v, z) = (s', v')$ and $\delta(s', \tau(v'), a) = (s'', \phi)$ for all $s \in S$, $v \in \mathbb{N}^k$, $z \in \Sigma^*$ and $a \in \Sigma$.

A *k-counter map automaton (k-CMA) $\mathcal{A}$ on the monoid presentation* $\mathcal{G} = \langle G|R\rangle$ is a tuple $(\mathcal{G}, k, S, s_0, \delta)$ where (G, R) is a finite presentation of $\mathcal{G}$ and (G, k, S, s_0, δ) is a k-counter map automaton on the alphabet G compatible with the monoid structure, *i.e.* satisfying $f(s_0, \mathbf{0}, zz_1) = f(s_0, \mathbf{0}, zz_2)$ for all $z \in G^*$ and $(z_1, z_2) \in R$. Its *mapping function* $g : \mathcal{G} \to S \times \mathbb{N}^k$ is defined ,for all $z \in \mathcal{G}$, as $g(z) = f(s_0, \mathbf{0}, z)$. Its *minimum (resp. maximum) counter function* $\min_c : \mathcal{G} \to \mathbb{N}$ (resp. $\max_c$) is defined for all $z \in \mathcal{G}$ as $\min_c(z) = \min_{i \in \mathbb{Z}_k} v_i$ (resp. $\max_c(z) = \max_{i \in \mathbb{Z}_k} v_i$) where $g(z) = (s, v)$. Two distinct elements $z, z' \in \mathcal{G}$ are *undistinguished* by $\mathcal{A}$ if $g(z) = g(z')$. An element $z \in \mathcal{G}$ is *discriminative* under $\mathcal{A}$ if $\min_c(z) = 0$. A connected subset Z of $\mathcal{G}$ is *independent* under $\mathcal{A}$ if $\{\min_c(z) \mid z \in Z\}$ is an infinite subset of $\mathbb{Z}^+$. Notice that a subset of $\mathcal{G}$ independent under $\mathcal{A}$ does not have any discriminative points under $\mathcal{A}$. The automaton is *periodic*, with period $z \in \mathcal{G} \setminus \{\epsilon\}$, if for all $z' \in \mathcal{G}$, $g(z') = g(z'z)$.

Lemma 1. *A k-CMA $\mathcal{A}$ on a group $\mathcal{G}$ is periodic if and only if two elements of $\mathcal{G}$ are undistinguished by $\mathcal{A}$.*

Proof. Let $\mathcal{A}$ be a k-CMA on a group $\mathcal{G}$. If it is periodic with period $z \in \mathcal{G}$ then ϵ and z are undistinguished. Conversely, if $z, z' \in \mathcal{G}$ are undistinguished then $z' - z$ is a valid period. ■

The *projector* $\pi_1 : S \times \mathbb{N}^k \to S$ is defined for all $s \in S$ and $v \in \mathbb{N}^k$ by $\pi_1(s, v) = s$. The *coloring* of $\mathcal{A}$ by $\varphi : S \to \Sigma$ is the mapping $c \in \Sigma^{\mathcal{G}}$ satisfying $c(z) = \varphi(\pi_1(g(z)))$ for all $z \in \mathcal{G}$. The *$\mathcal{G}$-k-map set* is the set of all colorings of $\mathcal{G}$ by all k-counter map automata. The *translated* of a coloring c, by a vector $z \in \mathcal{G}$, is the coloring $c_z \in \Sigma^{\mathcal{G}}$ defined for all $z' \in \mathcal{G}$ by $c_z(z') = c(zz')$.

Lemma 2. *Every $\mathcal{G}$-k-map set is closed under translation.*

Proof. Let c be a coloring of a k-CMA $(\mathcal{G}, k, S, s_0, \delta)$ by φ. Let $z \in \mathcal{G}$ be a vector and $m_z = \max_c(z)$. Let us consider functions $b : \mathbb{N} \to [0, \dots, m_z]$ and $t : \mathbb{N} \to \mathbb{N}$ defined for all $n \in \mathbb{N}$ by $b(n) = \min(m_z, n)$ and $t(n) = \max(0, n - m_z)$. Those two function can be naturally extended to $\mathbb{N}^k$. Let $S' = S \times [0, \dots, m_z]^k$ and $e : S \times \mathbb{N}^k \to S' \times \mathbb{N}^k$ be defined by $e(s, v) = ((s, b(v)), t(v))$ for all $(s, v) \in S \times \mathbb{N}^k$. Let $\mathcal{A}'$ be the k-CMA $(\mathcal{G}, k, S', s'_0, \delta')$ chosen such that, for all $z' \in \mathcal{G}$, its mapping function g' satisfies : $g'(z') = e(g(zz'))$ (in particular s' is the first component of $e(g(z))$). Straightforwardly, the translated c_z of c by z is the coloring of $\mathcal{A}'$ by $\varphi' : S \times [0, \dots, m_z]^k \to \Sigma$ defined, for all $s \in S$ and $t \in [0, \dots, m_z]^k$, as $\varphi'((s, t)) = \varphi(s)$. ■

Lemma 3. *Let c be a coloring of a k-CMA $\mathcal{A}$ on a group $\mathcal{G}$. Let $Z \subseteq \mathcal{G}$ be independent under $\mathcal{A}$. There exists a $\mathcal{G}$-0-map c' such that $c_{|Z} = c'_{|Z}$.*

Proof. Let $\mathcal{A}$ be a k-CMA $(\mathcal{G}, k, S, s_0, \delta)$. Let Z be a subset of $\mathcal{G}$ independent under $\mathcal{A}$. For all $s \in S$, let $Z_s = Z \cap g^{-1}\left(\{s\} \times \mathbb{N}^k\right)$. S being finite, there exists $s'_0 \in S$ such that $\left\{min_c(z) \mid z \in Z_{s'_0}\right\}$ is infinite. Let $\mathcal{A}'$ be the 0-CMA $(\mathcal{G}, 0, S, s'_0, \delta')$ where $\delta'(s, a) = \delta(s, +, a)$ for all $s \in S$ and $a \in G$. Let's first prove that $\mathcal{A}'$ is indeed a 0-CMA. Let $z \in G^*$ and $(z', z'') \in R$. Let $N = \max(|zz'|, |zz''|)$. By construction, there exists $z_N \in Z_{s'_0}$ such that $\min_c z_N > N$. Since no discriminative point is encountered on the path from z_N to $z_N zz'$, $f'(s'_0, zz') = s$ where $(s, n) = f(g(z_N), zz')$. The same holds for zz''. Let z_0 be any element of $Z_{s'_0}$, Z being independent, $c'(z) = c(z_0 z)$ for all $z \in Z$. By lemma 2, the translated c'_{z_0} of c' by z_0 is a $\mathcal{G}$-0-map equal to c on Z. ■

2 Map Automata on $\mathbb{Z}$

In the following, we denote by $\mathbb{Z}$ the one-dimensional grid

$$(\mathbb{Z}, +) = \langle l, r | lr = \epsilon, rl = \epsilon \rangle \quad ,$$

the presentation is embed with the canonical morphism $r = 1$.

2.1 Regular Colorings of $\mathbb{Z}$

For $i \in \mathbb{Z}$ and $p \in \mathbb{Z}^*$, let us denote as $p[i]$ the remainder of the division of p by i. On $\mathbb{Z}$, a periodic coloring c can be characterised by a finite pattern $u \in \Sigma^p$ such that for all $i \in \mathbb{Z}$, $c(i) = u_{i[p]}$. A coloring is *ultimately periodic* with period $p \in \mathbb{Z}^+$ and *defect* $k \in \mathbb{N}$, if for all element $i \in \mathbb{Z}, |i| > k$ implies $c(i+p) = c(i)$. Ultimately periodic colorings correspond to colorings which are periodic out of a finite support. They can also be characterised by three finite

words $u, v \in \Sigma^p$ and $w \in \Sigma^{2k+1}$ such that for all element $i \in \mathbb{Z}$, $c(i) = u_{i[p]}$ if $i < -k$, $c(i) = v_{i[p]}$ if $i > k$ and $c(i) = w_{i+k}$ otherwise. Notice that periodicity is a special case of ultimate periodicity.

2.2 Automata with no counter

Theorem 4. *A $\mathbb{Z}$-coloring is a $\mathbb{Z}$-0-map if and only if it is periodic.*

Proof. Let $g : \mathbb{Z} \to S$ be the mapping function of a 0-CMA $\mathcal{A}$. S being finite, there exists two elements undistinguished by $\mathcal{A}$. By lemma 1, the automaton is periodic.

Conversely, let c be a periodic $\mathbb{Z}$-coloring with period $p \in \mathbb{Z}^+$. Let $\mathcal{A}$ be the 0-CMA $(\mathbb{Z}, 0, \mathbb{Z}_p, 0, \delta)$ where $\delta(i, l) = i - 1$ and $\delta(i, r) = i + 1$. The coloring of $\mathcal{A}$ by $\varphi : i \mapsto c(i)$ is c. ■

2.3 Automata with 1 counter

Theorem 5. *A $\mathbb{Z}$-coloring is a $\mathbb{Z}$-1-map if and only if it is ultimately periodic.*

Proof. Let $g : \mathbb{Z} \to S \times \mathbb{N}$ be the mapping function of a 1-CMA $\mathcal{A}$. $\mathbb{Z}$-0-maps being $\mathbb{Z}$-1-maps, we can assume that g is one to one (*i.e.*, has no undistinguished elements). Thus, there exists $k \in \mathbb{N}$ such that $g^{-1}(S \times \{0\}) \subseteq [-k, k]$. By lemma 3, on both $]-\infty, -k[$ and $]k, +\infty[$, g is periodic and thus c is ultimately periodic.

Conversely, Let c be an ultimately periodic $\mathbb{Z}$-coloring with period $p \in \mathbb{Z}^+$ and defect $k \in \mathbb{N}$. Let $\mathcal{A}$ be the 1-CMA over set of states $[-k - p, k + p]$ whose mapping function is defined for all elements $i \in \mathbb{Z}$ by:

$$g(i) = \begin{cases} (i, 0) & \text{if } |i| \leq k + p \\ ((i - k)[p] + k, \lfloor (i - k)/p \rfloor) & \text{if } i > k + p \\ (i + k[p] - k - p, \lfloor (k - i)/p \rfloor) & \text{if } i < -k - p \end{cases}$$

The coloring of $\mathcal{A}$ by $\varphi : i \mapsto c(i)$ is c. ■

2.4 Automata with 2 counters

It is well known that finite automata with two counters can simulate any Turing machine (see Minsky [2]). Morita [3], improved the result proving that the simulation can be done in a reversible way. It is therefore no surprise that those machines can embed any computation of a Turing machine and encode any recursively enumerable language.

Theorem 6. *There exists a $\emptyset'$-complete $\mathbb{Z}$-2-map.*

Sketch of the proof. Let K be a $\emptyset'$-complete language containing 0 and $(n_i)_{i\in\mathbb{N}}$ be a computable enumeration without repetition of K satisfying $n_0 = 0$. There exists a one to one computable function that on input n_i computes n_{i+1}. Combining Morita construction [3] with techniques of [4], one can construct a reversible 2-counters machine with set of states $S_A \cup \{s_\alpha, s_\omega\}$ such that, for all $i \in \mathbb{N}$, starting from $(s_\omega, (n_i, 0))$, the machine eventually halts in configuration $(s_w, (n_{i+1}, 0))$. It is also possible to construct a reversible 2-counter machine with set of states $S_B \cup \{s_\alpha, s_\omega\}$ such that, for all $i \in \mathbb{N}$, starting from $(s_w, (n_i, 0))$, the machine halts in configuration $(s_0, (n_i, 0))$ after exactly $2n_i$ steps of computations. By making disjoint union of these two machines, one can construct 2-CMA $\mathcal{A}$ with set of states $S = S_A \cup S_B \cup \{s_\alpha, s_\omega\}$ sharing transitions of both machines. The transition function works by the applying 2-counter machine transition on generator r and reverse transition on generator l. Let c be the coloring of $\mathcal{A}$ by $\varphi : S \to \{0, 1\}$ defined for all $s \in S$ by $\varphi(s) = 1$ if and only if $s \in S_B$. The coloring c contains the factor $01^{2n}0$ if and only if $n \in K$. ■

3 Map Automata on $\mathbb{Z}^2$

In the following, we denote by $\mathbb{Z}^2$ the two-dimensional grid

$$(\mathbb{Z}^2, +) = \langle n, s, e, w | ns = \epsilon, sn = \epsilon, ew = \epsilon, we = \epsilon, ne = en \rangle \quad ,$$

the presentation is embed with the canonical morphism $e = \binom{1}{0}$ and $n = \binom{0}{1}$.

3.1 Regular Colorings of $\mathbb{Z}^2$

The case of $\mathbb{Z}^2$ is strongly linked with the previous lemma as the following case suggest.

Lemma 7. *Each line (resp. column) of a $\mathbb{Z}^2$-k-map is a $\mathbb{Z}$-k-map.*

Proof. Let c be a $\mathbb{Z}^2$-k-map. By definition, the restriction of c to $\{0\}\times\mathbb{Z}$ (resp. $\mathbb{Z}\times\{0\}$) is a $\mathbb{Z}$-k-map. By lemma 2, this result is valid for every line (resp. column). ■

An easy corollary is that periodic map automaton on $\mathbb{Z}^2$ acts as periodic copies of map automaton on $\mathbb{Z}$. To define regular coloring of $\mathbb{Z}^2$, we chose the approach presented in [1]. Simplest element is a biperiodic coloring. A *particle* is a coloring with one direction of periodicity and ultimate periodicity in every other direction. Let $\lhd_v(u,u')$ be the angular portion of the plane, on the right hand side of u, starting in position $v\in\mathbb{Z}^2$ and delimited by the vectors $u,u'\in\mathbb{Z}^2$. A *collision* as a coloring c characterised by a sequence of m vectors $(u_i)_{i\in\mathbb{Z}_m}$ such that for all $i\in\mathbb{Z}_m$, the coloring is u_i-periodic in the cone between u_{i-1} and u_{i+1} starting from u_i, *i.e.*, for all $i\in\mathbb{Z}_m$, and $z\in\lhd_{u_i}(u_{i-1},u_{i+1})$, $c(z+u_i)=c(z)$. The *ball* of radius r and center (x,y) is the set $[x-r,x+r]\times[y-r,y+r]$. When no center is specified, it implicitly refers to center $(0,0)$.

3.2 Automata with no counter

Theorem 8. *A $\mathbb{Z}^2$-coloring is a $\mathbb{Z}^2$-0-map if and only if it is biperiodic.*

Proof. Let $g:\mathbb{Z}\to S$ be the mapping function of a 0-CMA $\mathcal{A}$. S being finite, there exists two elements undistinguished by $\mathcal{A}$. Thus, there is only a finite number of lines or columns. By lemma 7 and theorem 4, each of them is periodic.

Let c be a biperiodic $\mathbb{Z}^2$-coloring with period $(m,0)$ and $(0,n)$ (such canonical periods always exists). Let $\mathcal{A}$ be the 0-CMA $(\mathbb{Z}^2,0,\mathbb{Z}_m\times\mathbb{Z}_n,(0,0),\delta)$ where $\delta((x,y),e)=(x+1,y)$ and $\delta((x,y),n)=(x,y+1)$. c is the coloring of $\mathcal{A}$ by $\varphi:\mathbb{Z}_m\times\mathbb{Z}_n\to\Sigma$ defined, for all $x\in\mathbb{Z}_m$ and $y\in\mathbb{Z}_n$, by $\varphi(x,y)=c(x,y)$. ■

3.3 Automata with 1 counter

Theorem 9. *A $\mathbb{Z}^2$-coloring is a $\mathbb{Z}^2$-1-map if and only if it is a particle.*

Proof. Let c be a $\mathbb{Z}^2$-1-map. Suppose that c is not periodic. By lemma 7, all lines of c are non periodic $\mathbb{Z}$-1-map and by lemma 3 each of them contains at least one discriminative point. $S\times\{0\}$ being finite, we reach a contradiction.

Thus, c is made of a finite number of lines (or columns) which are $\mathbb{Z}$-1-map. Since each of those map is ultimately-periodic, c is a particle.

Conversely, let c be a particle with period $u = (x, y)$ with $y > 0$ (this can always be achieved up to exchanging axes). Thus c consists of periodic repetitions of y ultimately periodic lines. Without loss of generality, we can assume that all the lines are $p \in \mathbb{Z}^+$ periodic with defect p with $p > x$. Thus, we can use the same construction as in proof of theorem 5 on the whole block of lines. Let $\mathcal{A}$ be a 1-CMA automaton over the set of states $[-k-p, k+p] \times [0, y-1]$ whose mapping function is defined for all element $(i, j) \in \mathbb{Z}^2$ by $g(i, j) = \tilde{g}(i + \lfloor j/y \rfloor\, x, j[y])$ where

$$\tilde{g}(\tilde{\imath}, \tilde{\jmath}) = \begin{cases} ((\tilde{\imath}, \tilde{\jmath}), 0) & \text{if } |\tilde{\imath}| \leq 2p \\ ((\tilde{\imath}[p] + p, \tilde{\jmath}), \lfloor \tilde{\imath}/p \rfloor - 1) & \text{if } \tilde{\imath} > 2p \\ ((\tilde{\imath}[p] - 2p, \tilde{\jmath}), \lfloor -\tilde{\imath}/p \rfloor - 1) & \text{if } \tilde{\imath} < -2p \end{cases}$$

The coloring of $\mathcal{A}$ by $\varphi : (i, j) \mapsto c(i, j)$ is c. ■

3.4 Automata with 2 counters

Theorem 10. *There exists a $\emptyset'$-complete $\mathbb{Z}^2$-2-map.*

Proof. It is possible to extend any $\mathbb{Z}$-k-map to a $\mathbb{Z}^2$-k-map by using identity function on e, w. Thus, existence of a $\emptyset'$-complete $\mathbb{Z}^2$-2-map induces existence of a $\emptyset'$-complete $\mathbb{Z}$-2-map. ■

Theorem 11. *Every collision is a $\mathbb{Z}^2$-2-map.*

Proof. Let c be a collision. For all $k \in \mathbb{N}$, let $w(k) = c_{|[-k,k]\times\{k\}}$ be the k-th *northern sphere word* of c. There exists a integer $K \in \mathbb{N}$ such that for all $k > K$, $w(k)$ is only included in cones (*i.e.*, avoid the central perturbation). The number of cones being finite, let $l \in \mathbb{N}$ be a multiple of vertical component of all the involved vectors. Then, for all $l' \in [0, \ldots, l-1]$, there exists $n \in \mathbb{N}$, $a(l', 0) \ldots a(l', n) \in \Sigma^*$ and $b(l', 0) \ldots b(l', n-1) \in \Sigma^*$ such that for all $k \in \mathbb{N}$, $kl > K$, $w(kl + l') = a(l', 0)b(l', 0)^k a(l', 1)b(l', 1)^k \ldots a(l', n-1)b(l', n-1)^k a(l', n)$. Notice that the set of constructed words $W = \{a(i, j) \mid i \in [0, \ldots, l'-1], i \in [0, \ldots, n]\} \cup \{b(i, j) \mid i \in [0, \ldots, l'-1], i \in [0, \ldots, n-1]\}$ is finite. Using similar techniques as previously, let us consider the partial mapping function g which maps any element to the corresponding letter in W. Moreover, local transition function is chosen such that, for any element in $w(kl + l')$, counter are equal to $(0, k)$ (resp. $(k, 0)$) if the corresponding letter

is in $a(l', 2i)$ (resp. $a(l', 2i+1)$) and $(i, k-i)$ (resp. $(k-i, i)$) if the letter is in the i-th repetition of the word $b(l', 2i)$ (resp. $b(l', 2i+1)$). One can note that g can be achieved by a local transition function. The last remaining problem is that g is, for now, only defined on the northern quarter of the plane.

Of course, the previous construction can be also achieved on all other quarter of the plane. What is left is to prove that those four construction can be chosen so that they match on boundaries. To do this, one has just to look at the diagonals (it is the only place where two or more constructions meet). Firstly, remark that value of the maximum counter depends on l which, up to taking a common multiple, can be set equal for all quarters. Secondly, the empty counter depends on the parity of number of particle involved. Since background can also be seen as particle, one can easily introduce "phantom" particles to get rid of this problem.

The last point is that the mapping resulting of the union of the four constructions is defined on all but the center which consists on a finite number of points. Up to introducing new states, on can extend this mapping function to the one of a 2-CMA on $\mathbb{Z}^2$. ■

Theorem 12. *Every aperiodic $\mathbb{Z}^2$-2-map is a collision.*

Proof. Let us take $\mathcal{A}$ a 2-CMA $(\mathbb{Z}^2, 2, S, s_0, \delta)$. $\mathcal{A}$ being aperiodic, its mapping function g is one to one. In a first step, let us prove that for any ball of radius r containing n discriminative points under $\mathcal{A}$, there is $n+1$ discriminative points under $\mathcal{A}$ in the ball of radius $r + |S| + 1$.

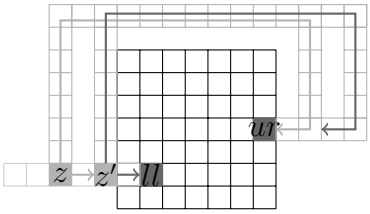


Fig. 1. Finding a new discriminative point

Let $\mathcal{B}$ be a ball of radius r containing n discriminative points. Assume that $\mathcal{B}'$ the ball of radius $r + |S| + 1$ does not contain any discriminative points. Let ul (resp. ur, ll) be one extremal upper-left (resp. upper-right, lower-left) discriminative point (see Fig. 1). Note that those points does not need to be distinct. Without loss of generality, one can assume that ur and ll have both the first counter empty. Let us consider the set $\{ll - (i, 0) \mid i \in \mathbb{N}\}$ of elements left of ll. S being finite, there exists two elements z and z' in $\mathcal{B}' \setminus \mathcal{B}$ such that $g(z) = (s, (a_0, a_1))$ and $g(z') = (s, (b_0, b_1))$ for some $s \in S$ and $a_0, a_1, b_0, b_1 \in \mathbb{Z}^+$. One can assume that z is the left one. There exist a path $r^i, i \in [0, \dots |S|]$ from z' to ll. Since the same path starting from z' does not encounter any disciminative point, $a_0 - (b_0 - 0) > 0$. By doing the same

reasoning on the path from z' to ur in $\mathcal{B}'$ avoiding $\mathcal{B}$, we can deduce that $b_0 - (a_0 - 0) > 0$ which gives a contradiction.

With this property on discriminative points, it is possible to establish a strong property on distribution of discriminative points and show that each discriminative point is located either in a fixed finite ball $\mathcal{B}$ or on an infinite half-line starting from this ball. Moreover, there is only a finite number of such half-lines.

Let $\mathcal{B}$ be the ball containing all discriminative points whose counters are both less than $(|S|+1)^3$. By iterating previous result on discriminative points, for each discriminative point z, there is at least $|S|+1$ discriminative points in the ball of radius $(|S|+1)^2$ centered around this point. Thus for each discriminative point, there exists, a path of length less than $(|S|+1)^3$ containing $|S|+1$ discriminative points. Either two discriminative points have distinct empty counter which implies that non empty counter of z is less the $(|S|+1)^3$, or there exists two points z_a and z_b such that $g(z_a) = (s, (a, 0))$ and $g(z_b) = (s, (b, 0))$ for some $s \in S$ and $b > a \in \mathbb{N}^+$. Let $z \in G^*$ such that $z_a z = z_b$. One can check that $g(z_a z^n) = (s, (a - n(b-a), 0))$ for all n such that $a - (b-a)^n > (|S|+1)^3$. Thus z_a is on a half-line starting from an elements in $\mathcal{B}$. Slope of this half-line depends only on the point in the ball and $b - a < (|S|+1)^3$ which only leave a finite number of possibilities.

The last intermediate result needed is that any point $z \in \mathbb{Z}^2$ such that $min_c z < N$ has one discriminative point under $\mathcal{A}$ at distance at most $N(|S|+1)$. To prove this, let $\mathcal{B}$ be the ball of center $z \in \mathbb{Z}^2$ and of radius $|S|$. This ball contains two distinct points $z_a, z_b \in \mathbb{Z}^2$ whose mapping has the same state. Applying same reasoning as previously allow to obtain the desired result. A useful corollary of this result is that any connected component between two consecutive (but not parallel) half-lines of discriminative points is independent since it contains ball of arbitrary size.

To conclude the proof, let us show that the map is a collision chracterised by the sequence $u \in \mathbb{Z}^{2p}$ of half-lines ordered by slope. Since all half-lines start inside a finite ball, there exist a element $k \in \mathbb{N}$ such that all discriminative points in the cone $\lhd_{ku_i}(ku_{i-1}, ku_{i+1})$ are on a half-line of vector u_i for all $i \in \mathbb{Z}_p$. In this cone, we have element of the half-line (which are by construction u_i periodic) and elements of connected components between two consecutive half-lines. Those connected component are independent by the previous corollary. By lemma 3, they are bi-periodic and thus also ku_i periodic for some $k \in \mathbb{N}^+$.

■

References

[1] N. Ollinger, G. Richard, Collisions and their catenations: Ultimately periodic tilings of the plane, in: Proceedings of the Fifth IFIP Int. Conf. on TCS, Vol. 273, Springer, 2008, pp. 229–240.

[2] M. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, Englewoods Cliffs, 1967.

[3] K. Morita, Universality of a reversible two-counter machine, Theor. Comput. Sci. 168 (2) (1996) 303–320.

[4] J. Kari, N. Ollinger, Periodicity and immortality in reversible computing, (*MFCS 2008, to appear*).

ANNEXE

B INDÉCIDABILITÉS, MACHINES ET APÉRIODICITÉS

Recueil des articles les plus représentatifs concernant le second chapitre du mémoire. Le lecteur y trouvera les principaux résultats des deux principales sections : pavabilité du plan et immortalité des machines ; mais aussi un résultat d'indécidabilité en combinatoire des mots qui illustre le codage par raffinements successifs d'un modèle de calcul ad hoc. *Lorsqu'elles existent, nous avons choisi de présenter les versions longues (éventuellement soumises mais non encore acceptées) plutôt que les communications de conférence correspondantes.*

Sommaire

B.1 Two-by-two Substitutions Systems and the Undecidability of the Domino Problem

Version rapporteur de [C4], un article présenté à CiE'2008 proposant une nouvelle construction et technique de preuve pour l'indécidabilité de la pavabilité du plan.

Two-by-two Substitution Systems and the Undecidability of the Domino Problem

Nicolas Ollinger

Laboratoire d'informatique fondamentale de Marseille (LIF)
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France,
Nicolas.Ollinger@lif.univ-mrs.fr

Abstract. Thanks to a careful study of elementary properties of two-by-two substitution systems, we give a complete self-contained elementary construction of an aperiodic tile set and sketch how to use this tile set to elementary prove the undecidability of the classical Domino Problem.

Introduction

The *Domino Problem* is the following simple problem: given a finite set of tiles, copies of the unit square with colored edges, decide if it is possible to tile the whole euclidian plane using as many copies of each tile as you need ensuring that tiles colors match along edges, without scaling or rotating the tiles. This problem was first described by Wang to study a particular syntactical restricting of the Entscheidungsproblem (for a proof of the logical problem without using the Domino Problem and an explanation of the field, see [6]). The Domino Problem turns out to be undecidable, as it was proved in 1964 by Berger [1, 2], a student of Wang. The undecidability of the Domino Problem as since been used outside its original realm, providing a valuable tool to prove undecidability results, see for example the results of Kari [7] on cellular automata.

The historical proof, as found in [1], is very technical. One technical difficulty of the proof is the involvement of aperiodic tile sets: set of tiles that only admit aperiodic tilings. Several authors worked both to ameliorate the proof of the Domino Problem and to construct simpler aperiodic tile sets. The most quoted proof is certainly the one from Robinson [12], the first proof involving substitutions is the one of Mozes [10]. For an historical survey of the field, the reader might consult [5] and [11].

Recently, several authors have been independently interested into providing new proofs of the undecidability of the Domino Problem: Durand, Levin, and Shen [3] revisited the classical Robinson proof technic introducing new tools based on substitutions; Kari [8] proposed a completely new proof technic reducing the Domino Problem to the immortality problem rather than on the halting problem; Durand, Romashchenko, and Shen [4] proposed another new proof technic based on Kleene's fixed point theorem.

In the present paper, we give a complete self-contained elementary construction of an aperiodic tile set by combining tools from [12, 10, 3] with a careful

study of two-by-two substitution systems. Our claim is that this proof both explains where the tile set comes from and why it works. Moreover, the number of tiles (104) is a reasonable compromise between classically big tile sets (more than 16000 tiles in [12, 3]) and very involved small tile sets (see [5] for details on the competition). Furthermore, the tile set as a nice property to be easily extendable to code any substitution, leading to a new and shorter proof of the undecidability of the Domino Problem sketched in 4.

1 Two-by-two Substitution Systems

A *pattern* $\mathcal{P}$ is a subset of the discrete plane $\mathbb{Z}^2$. The *translation* $\mathcal{P}+u$ of a pattern $\mathcal{P}$ by a vector $u \in \mathbb{Z}^2$ is the pattern $\{z+u \mid z \in \mathcal{P}\}$. Let $\boxplus^i$ denote for all $i \in \mathbb{N}$ the pattern $\left\{x \in \mathbb{Z} \mid 0 \leqslant x < 2^i\right\}^2$: the square of size 2^i with south-west corner at $\binom{0}{0}$. The two-by-two square $\boxplus^1$ is abbreviated as $\boxplus$. The two-by-two *scaling* $\square(\mathcal{P})$ of a pattern $\mathcal{P}$ is the pattern $\{2z+c \mid z \in \mathcal{P}, c \in \boxplus\}$.

Let Σ denote a finite set, or *alphabet*. A *coloring* $\mathcal{C} : \mathcal{P} \to \Sigma$ is a covering of a pattern $\mathcal{P}$, the support of $\mathcal{C}$ denoted as $\mathrm{Sup}(\mathcal{C})$, by letters of Σ. A *subcoloring* $\mathcal{C}'$ of a coloring $\mathcal{C}$ is a restriction of this coloring, formally $\mathcal{C}' = \mathcal{C}_{|\mathrm{Sup}(\mathcal{C}')}$. The *translation* $u \cdot \mathcal{C}$ of a coloring $\mathcal{C}$ by a vector $u \in \mathbb{Z}^2$ is the coloring with support $\mathrm{Sup}(\mathcal{C})+u$ satisfying $u \cdot \mathcal{C}(z+u) = \mathcal{C}(z)$ for all $z \in \mathrm{Sup}(\mathcal{C})$. A coloring $\mathcal{C}$ *occurs* in a coloring $\mathcal{C}'$, denoted as $\mathcal{C} \prec \mathcal{C}'$, if a translation of $\mathcal{C}$ is a subcoloring of $\mathcal{C}'$.

A coloring $\mathcal{C}$ is *periodic*, with period vector $p \in \mathbb{Z}^2$, if, for all $z \in \mathrm{Sup}(\mathcal{C})$, if $z+p \in \mathrm{Sup}(\mathcal{C})$ then $\mathcal{C}(z+p) = \mathcal{C}(z)$. An *aperiodic* coloring is a coloring admiting no non-trivial period (*i.e.* other than the trivial period 0). A set of coloring is *aperiodic* if it is not empty and all its colorings are aperiodic.

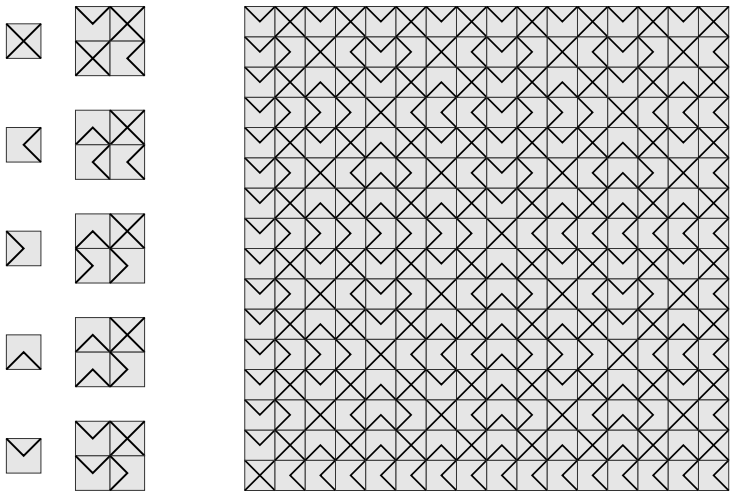
Let X be the set of colorings with support $\mathbb{Z}^2$. Endow X with the product topology of the discrete topology on Σ. This topology is compatible with the metric d defined for all colorings $\mathcal{C}, \mathcal{C}' \in X$ by $d(\mathcal{C}, \mathcal{C}') = 2^{-\min\left\{|z|,\, \mathcal{C}(z) \neq \mathcal{C}'(z)\right\}}$. Such topology is compact and perfect. A subset of X both topologically closed and closed by translations is a *subshift* from symbolic dynamics [9].

A two-by-two *substitution system* is a pair (Σ, s) where Σ is a finite alphabet and $s : \Sigma \to \Sigma^{\boxplus}$ is called the *substitution rule*. The local rule s is extended to a *global rule* $S : \Sigma^{\mathcal{P}} \to \Sigma^{\square(\mathcal{P})}$ mapping colorings into colorings by:

$$\forall z \in \mathcal{P}, \forall c \in \boxplus, \quad S(\mathcal{C})(2z+c) = s(\mathcal{C}(z))(c) \quad .$$

The restriction of the global rule to X is a continuous map. The global rule weakly commutes with translations: $S(u \cdot \mathcal{C}) = 2u \cdot S(\mathcal{C})$ for all vector $u \in \mathbb{Z}^2$ and all coloring $\mathcal{C}$. The *i-level* image of a letter $a \in \Sigma$ by s is the coloring $S^i(a)$ with support $\boxplus^i$.

Example 1. Figure 1 depicts a variation on the classical chair two-by-two substitution. This substitution will reappear later in this paper. □



(a) rule s (b) 4-level image $S^4(\boxtimes)$

Fig. 1: A sample two-by-two substitution with 5 letters

The literature provides several different methods to extend substitutions to colorings of the whole plane. A classical one is to consider the set X_s of colorings such that each of their finite subcolorings $\mathcal{C}$ occurs at some level i, that is there exists $a \in \Sigma$ such that $\mathcal{C} \prec S^i(a)$.

In this paper[1], we prefer to take a more dynamical point of view by considering the limit set Λ_S, the intersection $\bigcap_{n\in\mathbb{N}} \Lambda_S^n$ of a decreasing sequence of nonempty subshifts. Let $\Lambda_S^0 = X$ and $\Lambda_S^{n+1} = \{u \cdot S(\mathcal{C}) \mid \mathcal{C} \in \Lambda_s^n, u \in \boxplus\}$, for all $n \in \mathbb{N}$. As S weakly commutes with translations, Λ_S^n is precisely the closure by translation of the compact set $S^n(X)$. Notice that, depending on s, the sets X_s and Λ_S might be different.

Example 2. Consider the constant substitution on two letters $(\{a, b\}, \lambda x.\lambda z.x)$. The set X_s contains 2 elements, but Λ_S contains infinitely many elements (the closure by translation of 16 elementary elements). □

An *history* for a coloring $\mathcal{C} \in X$ is a sequence $(\mathcal{C}_i, u_i) \in (X \times \boxplus)^{\mathbb{N}}$ such that $\mathcal{C}_0 = \mathcal{C}$ and $\mathcal{C}_i = u_i \cdot S(\mathcal{C}_{i+1})$, for all $i \in \mathbb{N}$.

Proposition 1. *The set Λ_S is precisely the set of colorings admitting histories.*

Proof. Consider an history $(\mathcal{C}_i, u_i) \in (X \times \boxplus)^{\mathbb{N}}$. As $\mathcal{C}_0$ is a translation of $S^i(\mathcal{C}_i)$ for all $i \in \mathbb{N}$, straightforwardly $\mathcal{C}_0 \in \Lambda_S$. Conversely, let $\mathcal{C}$ be a coloring in Λ_S. By construction, one can find $(\mathcal{C}_i, u_i) \in (X \times \boxplus)^{\mathbb{N}}$ such that $\mathcal{C}_0 = \mathcal{C}$ and $\left(\sum_{j=0}^{i-1} 2^j u_j\right) \cdot S^i(\mathcal{C}_i) = \mathcal{C}$. By compacity of X, one can extract from the iterated

[1] in fact, even when claiming to construct tilings coding X_s, all the constructions we know about really code Λ_S and can do it for all s, not only s such that $X_s = \Lambda_S$.

images by S of subcolorings of the family $(\mathcal{C}_i)$ a new family of colorings $(\mathcal{C}'_i)$ such that $\mathcal{C}'_i = u_i \cdot S(\mathcal{C}'_{i+1})$, for all $i \in \mathbb{N}$. ■

Informally, to code elements of Λ_S using only local rules, one can code a whole history and ensure that the validity of the history is locally checkable. One way to achieve that is to ensure that part of the *story* at each position is locally available. A story simply explains the value of at a particular position according to an history: it is the sequence of substitution rules applied to obtain the value at that particular position.

More formally, the *story* at position $z \in \mathbb{Z}^2$ for an history $(\mathcal{C}_i, u_i) \in (X \times \boxplus)^{\mathbb{N}}$ is the sequence $(a_i, v_i) \in (\Sigma \times \boxplus)^{\mathbb{N}}$ such that, for all $i \in \mathbb{N}$, $a_i = s(a_{i+1})(v_i)$ and $a_i = \mathcal{C}_i(z_i)$ where $z_i \in \mathbb{Z}^2$ is the only position such that z is an element of the pattern $\mathcal{P}_i = \boxplus^i - \sum_{j=0}^{i-1} 2^j u_j - 2^i z_i$. Notice that the subcoloring of $\mathcal{C}$ of support $\mathcal{P}_i$ is a translation of $S^i(a_i)$.

Every story is computable from the story of any among two of its four neighbors, most of the time from any of them. Let $\left(a_i, \binom{x_i}{y_i}\right) \in (\Sigma \times \boxplus)^{\mathbb{N}}$ be a story at position $z \in \mathbb{Z}^2$, the story $(b_i, u_i) \in (\Sigma \times \boxplus)^{\mathbb{N}}$ at position $z + \binom{1}{0}$ can be constructed as follows. Let $k \in \mathbb{N}$ be the smallest k such that $x_k = 0$. Let $u_k = \binom{1}{y_k}$. For all $i < k$, let $u_i = \binom{0}{y_i}$. For all $i > k$, let $u_i = \binom{x_i}{y_i}$ and $b_i = a_i$. For all $i \leqslant k$ let b_i be such that $b_i = s(b_{i+1})(u_i)$. This procedure will always produce the story for position $z + \binom{1}{0}$, but when k is not defined. Stories at positions $z + \left\{-\binom{1}{0}, \binom{0}{1}, -\binom{0}{1}\right\}$ can be defined symmetrically.

Proposition 2. *Every history can be reconstructed from 1,2, or 4 of its stories.*

Proof. By construction, an history is completely defined by the set of all its stories. Consider any story $(a_i, \binom{x_i}{y_i})$ of a given history. Four different cases may occur depending on both sequences (x_i) and (y_i). If none of them is ultimately constant, the history can be reconstructed by reconstructing the stories of each position of the plane. If exactly one of them is ultimately constant, only half a plane of stories can be reconstructed and two different stories, with different ultimate constants, are needed. If both sequences are ultimately constant, only a quarter of the plane of stories can be reconstructed and four different stories, with different ultimate constants, are needed. ■

A substitution is *aperiodic* if its limit set Λ_S is aperiodic. A substitution is *unambiguous* if, for every coloring $\mathcal{C}$ from its limit set Λ_S, there exists a unique coloring $\mathcal{C}'$ and a unique translation $u \in \boxplus$ satisfying $\mathcal{C} = u \cdot S(\mathcal{C}')$. Notice that the injectivity of the local rule is not sufficient to enforce unambiguity. Every unambiguous substitution admits a unique history.

Proposition 3. *Every unambiguous substitution is aperiodic.*

Proof. Let s be an unambiguous substitution. Let us assume that s is not aperiodic. Let p be the smallest non-trivial period of a coloring in the limit set of s for

the maximum norm. Let $\mathcal{C} \in \Lambda_S$ be p-periodic. Let u and $\mathcal{C}'$ satisfy $\mathcal{C} = u \cdot S(\mathcal{C}')$. By construction, $(p+u) \cdot S(\mathcal{C}')$ is equal to $\mathcal{C}$. As s is unambiguous, p has to be a multiple of 2, $p = 2p'$, so that $(p+u) \cdot S(\mathcal{C}') = u \cdot S(p' \cdot \mathcal{C}')$ and $\mathcal{C}' = p' \cdot \mathcal{C}'$. Thus $\mathcal{C}' \in \Lambda_S$ is p'-periodic and p' is smaller than p, contradicting our hypothesis. ■

Example 3. The substitution on Figure 1 is unambiguous thus aperiodic. □

A syntactical way to enforce unambiguity of a substitution s, as used in [3], is to ensure that it is injective and that one of the four projectors $s_i : a \mapsto s(a)(i)$ has an image disjoined from the images of the other three.

2 Tilings

A *domino relation* $\mathcal{R} \subseteq Y \times Y$ over a finite set Y satisfies the domino property :

$$\forall a, b, c, d \in Y^4, \quad a\mathcal{R}c \wedge a\mathcal{R}d \wedge b\mathcal{R}d \rightarrow b\mathcal{R}c \tag{1}$$

The color set associated to a domino relation $\mathcal{R}$ is the set of equivalence classes of the equivalence relation $\sim_{\mathcal{R}}$ defined on Y^2 for all $a, b, c, d \in Y$ by $(a, c) \sim_{\mathcal{R}} (b, d)$ if $a\mathcal{R}c \wedge a\mathcal{R}d \wedge b\mathcal{R}d$. The right color of an element $a \in Y$ is the color $|a\rangle$ such that there exists b satisfying $(a, b) \sim_{\mathcal{R}} |a\rangle$. Symmetrically, the left color of an element $b \in Y$ is the color $\langle b|$ satisfying $(a, b) \sim_{\mathcal{R}} \langle b|$. Straightforwardly, for all $a, b \in Y$, $a\mathcal{R}b$ if and only $|a\rangle = \langle b|$.

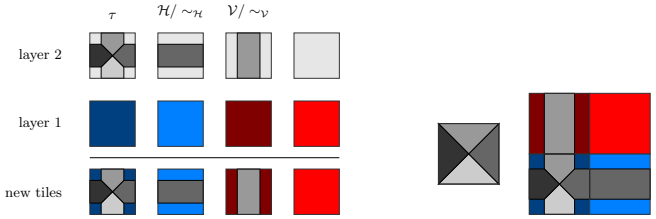
Tilings correspond to the extension of subshifts of finite type (SFT) [9] to $\mathbb{Z}^2$: colorings of the plane satisfying a finite set of local constraints. Several definitions of tiling constraints are possible. In this paper, we focus on so called Wang tiles, but instead of using the classical definition by colors, we use domino relations to simplify the discussions. A *tile set* τ is a triple $(T, \mathcal{H}, \mathcal{V})$ where T is a finite alphabet of tiles, $\mathcal{H}$ and $\mathcal{V}$ are domino relations over T. A tile set is *degenerated* if two tiles $a, b \in T$ define the same quadruple of colors $(|a\rangle_{\mathcal{V}}, |a\rangle_{\mathcal{H}}, \langle a|_{\mathcal{V}}, \langle a|_{\mathcal{H}})$ and $(|b\rangle_{\mathcal{V}}, |b\rangle_{\mathcal{H}}, \langle b|_{\mathcal{V}}, \langle b|_{\mathcal{H}})$. A pair of tiles $(a, b) \in T^2$ *matches horizontally* if $a\mathcal{H}b$, *matches vertically* if $a\mathcal{V}b$. A *tiling* $\mathcal{T} \in X$ is a coloring satisfying the tiling constraints: for all $z \in \mathbb{Z}^2$, the pair $\left(\mathcal{T}(z), \mathcal{T}\left(z + \binom{1}{0}\right)\right)$ matches horizontally and the pair $\left(\mathcal{T}(z), \mathcal{T}\left(z + \binom{0}{1}\right)\right)$ matches vertically. The set of tilings X_τ of a tile set τ is a subshift (of finite type). A tile set is *aperiodic* if its set of tilings is aperiodic.

Example 4. The tile set $\tau_0 = \left(\boxplus, \left\{(u, v) \middle| |v - u| = \binom{1}{0}\right\}, \left\{(u, v) \middle| |v - u| = \binom{0}{1}\right\}\right)$ admits 4 tilings: the limit set of the simple substitution $\lambda x.\lambda z.z$. □

A tile set $(T', \mathcal{H}', \mathcal{V}')$ *codes* a tile set $(T, \mathcal{H}, \mathcal{V})$, according to a *coding rule* $t : T \rightarrow T'^{\boxplus}$ if t is injective and $X_{\tau'} = \{u \cdot t(\mathcal{C}) | \mathcal{C} \in X_\tau, u \in \boxplus\}$.

Example 5. A simple coding scheme is depicted on figure 2. Given a tile set $(T, \mathcal{H}, \mathcal{V})$, the coding tile set is constructed as a layered tile set : a synchronized product of several layers, upper layers being constrained by lower layers. Layer 1

is the tile set from example 4. Layer 2 is constrained according to layer 1 : on top of $\binom{0}{0}$ stack elements of T; on top of $\binom{1}{0}$ stack elements of $\mathcal{H}/\sim_{\mathcal{H}}$; on top of $\binom{0}{1}$ stack elements of $\mathcal{V}/\sim_{\mathcal{V}}$; on top of $\binom{1}{1}$ is an empty element. The matching rules for layer 2 are simple : a tile in T is required to match horizontal and vertical colors of its four neighbor tiles thus propagating its colors to the next $\binom{0}{0}$ tile. The depicted coding rule t maps every tile a as follows. On layer 1, $t(a)$ is the identity map. On layer 2, $t(a)$ puts a on top of $\binom{0}{0}$, $|a\rangle_{\mathcal{H}}$ on top of $\binom{1}{0}$ and $|a\rangle_{\mathcal{V}}$ on top of $\binom{0}{1}$. □



(a) coding tile set (b) coding rule

Fig. 2: A sample coding scheme

A tile set $(T, \mathcal{H}, \mathcal{V})$ *codes* a substitution $s : T \to T^{\boxplus}$ if it codes itself according to the coding rule s.

Proposition 4. *A tile set both admitting a tiling and coding an unambiguous substitution is aperiodic.*

Proof. Let $(T, \mathcal{H}, \mathcal{V})$ be a tile set admitting a tiling and coding an unambiguous substitution $s : T \to T^{\boxplus}$. By construction, every tiling is the translated image of a tiling by s, thus $X_\tau \subseteq \Lambda_S$. As X_τ is not empty and Λ_S is aperiodic, X_τ is aperiodic. ■

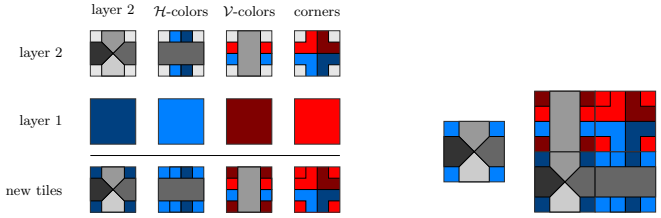
3 An Aperiodic Tile Set of 104 Tiles

To apply proposition 4, we construct a fixed point[2] of a coding scheme in the spirit of the one described in example 5 (this particular coding scheme will not help: the new tile set is always strictly larger than the original one).

One can refine the scheme, as depicted on Figure 3: as the coding scheme generates a layered tile set, one might forget about layer 1 on top of $\binom{0}{0}$ and code it all around on wires in $\binom{1}{0}$, $\binom{0}{1}$, and $\binom{1}{1}$. On top of $\binom{1}{1}$, 4 different corners can occur corresponding to the 4 different corners in τ_0. On top of $\binom{0}{1}$, 4 possible pairs

[2] the way we use fixed points with tilings in this paper, whereas sharing similarities with the approach in [4], is less sophisticated.

of wires propagates vertically crossing the $\mathcal{H}$-colors that propagates horizontally. The case of $\binom{1}{0}$ is symmetrical. The new matching rule on layer 2 still requires to match colors but also wires between $\binom{1}{1}$ and its neighbors.



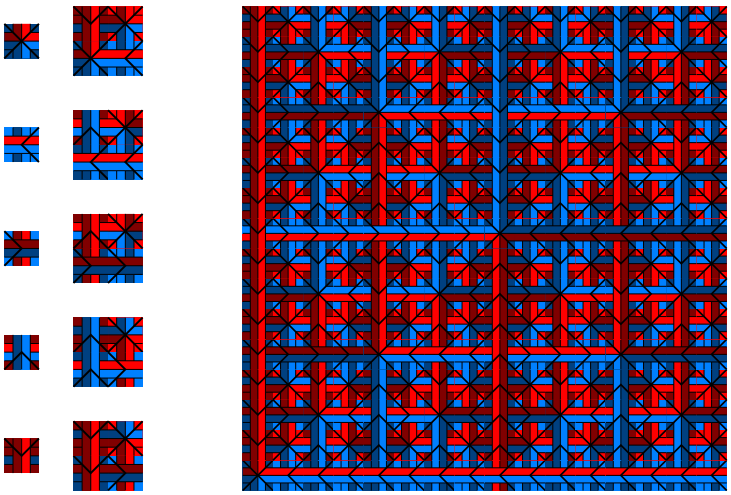
(a) coding tile set (b) coding rule

Fig. 3: A second coding scheme

A priori, this new coding scheme cannot be applied to every tile set. In a coded tiling, each tile is coded both by its layer 2 component on top of $\binom{0}{0}$ and its layer 1 component by a square wire around neighbors tiles. The only constraints between layer 1 and layer 2 values are checked on $\binom{0}{1}$ and $\binom{1}{0}$ restricting possible $\mathcal{H}$-colors and $\mathcal{V}$-colors for a given layer 1 tile. The coding scheme can only be applied to tile sets closed with respect to this property: if a layer 2 tile a_2 exists and its four $\mathcal{H}$-colors and $\mathcal{V}$-colors are compatible with a given layer 1 tile a_1, the tile (a_1, a_2) occurs in the tile set.

The new coding scheme cannot be iterated: the coded tile set does not satisfy the closure hypothesis. This can be corrected by adding one bit of information on the wire pair edges of each tile to indicate on which side of the edge one can find the nearest corner: on top of $\binom{1}{1}$ corners are always inside; on top of $\binom{1}{0}$, the nearest corners are outside for both vertical edges; on top of $\binom{0}{1}$, the nearest corners are outside for both horizontal edges. The matching rule has to enforce that both sides of a wire pair edge agree on the direction. With these new bits of information, the coding scheme preserves the closure property and can be iterated. Moreover, it is not strictly increasing and admits a fixed point: a tile set τ of 104 tiles, the coding substitution of which is depicted on figure 4. The bits on the edges are considered as a third layer, an inside edge being represented by a V shape pointing the center of the tile: exactly five tiles occur on layer 3, the letters of figure 1.

Before proving the aperiodicity of τ, let us first describe it more precisely. The tile set has three layers. Layer 1 is τ_0. Layer 2 consists of X, H and V tiles (see below) transmitting pairs of wires from edge to edge, each wire being colored by an element of τ_0 so that on edges, only pairs satisfying $\mathcal{H}_{\tau_0}$ and $\mathcal{V}_{\tau_0}$ are valid. The matching rule on layer 2 is to match the colors of the facing wires. Layer 3 consists of the five letters from figure 1. The matching rule on layer 3 is to agree on direction (exactly one V shape along each edge pointing in one of the two directions). Only the following synchronizations between layers occur inside τ:



(a) principle (b) 4-level image S^4( )

Fig. 4: a 104 tiles aperiodic tile set τ coding an unambiguous substitution

8 X tiles. layer 2 with layer 1 $\binom{0}{0}$ or $\binom{1}{1}$ and layer 3 equal to .
48 H tiles. layer 2 with layer 1 $\binom{0}{0}$ or $\binom{1}{0}$ and layer 3 equal to or .
48 V tiles. layer 2 with layer 1 $\binom{0}{0}$ or $\binom{0}{1}$ and layer 3 equal to or .

There is only 48 tiles of type H and V (instead of 64) because the propagating direction of their layer 3 has to satisfy the wire color constraint: if one of the orthogonal wires is colored $\binom{1}{1}$, the direction has to point inside the $\binom{1}{1}$ boundary.

The associated substitution rule s transforms a tile a as follows. In position $\binom{0}{0}$, layer 1 is $\binom{0}{0}$ and layers 2 and 3 are the same as for a. In position $\binom{1}{1}$, there is an X tile with layer 1 $\binom{0}{0}$ and bottom-left wire color equal to the layer 1 of a. In position $\binom{1}{0}$, there is an H tile with layer 1 $\binom{1}{0}$: it propagates the wire colors of both its neighbors and points in the same direction as the right edge of the layer 3 of a. In position $\binom{0}{1}$, there is a V tile with layer 1 $\binom{0}{1}$: it propagates the wire colors of both its neighbors and points in the same direction as the top edge of the layer 3 of a. Notice that the image of a tile is always defined as the wire color constraint is always satisfied.

Theorem 1. *The tile set τ is aperiodic.*

Proof. The tile set τ admits at least one tiling. Consider the substitution s: the four inside edges of each image of a tile by the substitution satisfy the matching rules. Moreover, if two tiles a and b match either horizontally or vertically, $s(a)$ and $s(b)$ will still match in the same direction. Therefore, $\Lambda_S \cap X_\tau \neq \emptyset$.

The substitution s is unambiguous: it is clearly injective and all the s_i projectors have disjoined images.

The tile set τ codes s. Let $\mathcal{T}$ be a tiling of τ. As layer 1 admits only 4 tilings, there exists $u \in \boxplus$ such that, for all $v \in \boxplus$, $u \cdot \mathcal{T}(v)$ has layer 1 v. Thus, $u \cdot \mathcal{T}$ has an X tile in $\binom{1}{1}$, an H tile in $\binom{1}{0}$, and a V tile in $\binom{0}{1}$. Thanks to the wire color constraint, $u \cdot \mathcal{T}_{|\boxplus}$ has to be the image of a tile by s. Repeating the argument on the $2\mathbb{Z}^2$ translations of $\mathcal{T}$, one conclude that $\mathcal{T}$ is the translated image of coloring by s. Moreover, this coloring is a valid tiling: as the information is propagated on layer 2 and 3 by H and V tiles, if two tile images $s(a)$ and $s(b)$ match, then a and b match. Therefore, as the image of a tiling by s is also a tiling, $X_\tau = \{u \cdot s(\mathcal{C}) | \mathcal{C} \in X_\tau, u \in \boxplus\}$.
As τ admits a tiling and codes an unambiguous substitution, it is aperiodic. ■

The tile set τ somehow uses its layers 1 and 2 to draw infinitely many infinite grids to code an history of the substitution of figure 1: the set of layer 3 of tilings is the limit set of the chair substitution. When comparing τ to aperiodic tile sets of the literature, the author found the following facts. The PhD dissertation of Berger [1] contains an aperiodic 104 tile set not found in the AMS memoir [2]. Berger's tile set also has three layers, the first two being isomorphic to the one used here... but the third layer is different leading to a more delicate proof and a different set of tilings! Let now τ' be the modified tile set where the colors $\binom{0}{0}$, $\binom{1}{0}$ and $\binom{0}{1}$ are merged in one unique color. The tile set τ' admits more tilings than just the repainted tilings of τ, it has 56 tiles: it is the tile set of Robinson from [12], the aperiodicity proof of which is rather technical.

4 Enforcing any Substitution

The tile set τ might be slightly modified to enforce the limit set of any substitution system s': the idea is to use $\binom{1}{1}$ squares of all sizes to propagate the history of an element of $\Lambda_{S'}$. The transformed tile set $\tau(s')$ is constructed from τ by replacing $\boxplus$ with $\boxplus \times \Sigma$ on layer 1 and on the wires of layer 2. The matching rule is extended so that letters have to be equal on layer 1 on the four edges between $\binom{0}{0}$ and $\binom{1}{0}$, $\binom{0}{0}$ and $\binom{0}{1}$, $\binom{1}{0}$ and $\binom{1}{1}$, and $\binom{0}{1}$ and $\binom{1}{1}$. Then, both X tile and V tile wire colors are constrained. For any X tile, let a be the letter on the $\binom{1}{1}$ wire and b be the letter on layer 1, the constraint is $b = s'(a)(u)$ where u depends on the position of the $\binom{1}{1}$ wire ($\binom{0}{0}$ for top-right, $\binom{1}{0}$ for bottom-right, $\binom{0}{1}$ for top-left, $\binom{1}{1}$ for bottom-left). For any V tile with two $\binom{1}{1}$ wire, let a be the letter on the vertical $\binom{1}{1}$ wire and b the letter on the horizontal one, the constraint is $b = s'(a)(u)$ where $u = \binom{x}{y}$ depends on the positions of both wires ($x = 0$ for right, $x = 1$ for left, $y = 0$ for bottom, $y = 1$ for top). Let π map every tile of $\tau(s')$ to $s'(a)(u)$ where a and u are the letter and the value of $\boxplus$ on layer 1.

Theorem 2. *Let s' be any substitution system. The tile set $\tau(s')$ enforces s': $\pi\left(X_{\tau(s')}\right) = \Lambda_{S'}$.*

Sketch of the proof. Every tiling of $\tau(s')$ codes an history of S' and every history of S' can be encoded into a tiling of $\tau(s')$.

Corollary 1 (Berger, 1964 [1]). *The Domino Problem is undecidable.*

Sketch of the proof. Consider the following 6 letters substitution $\mathfrak{s}$:

$$T \mapsto \begin{matrix} t & t \\ t & t \end{matrix} \quad V \mapsto \begin{matrix} v & v \\ v & v \end{matrix} \quad H \mapsto \begin{matrix} h & h \\ h & h \end{matrix} \quad t \mapsto \begin{matrix} T & V \\ H & T \end{matrix} \quad v \mapsto \begin{matrix} T & V \\ H & V \end{matrix} \quad h \mapsto \begin{matrix} T & V \\ H & H \end{matrix}$$

Consider the letter T as a place for a tile  , the letter H as a horizontal color transmission path  , and the letter V as a vertical color transmission path  . Every coloring $\mathcal{C}$ of the limit set $\Lambda_{\mathfrak{S}}$ containing letters H, V, T has the following property: for all $i \in \mathbb{N}$, a $\boxplus^i$ square of T letters, bordered horizontally by V letters and vertically by H letters, eventually spaced by H and V letters providing color transmission, occurs in $\mathcal{C}$. To prove the undecidability of the Domino Problem, recursively construct for every Turing machine a tile set with two layers: on layer 1 put $\tau(\mathfrak{s})$ with the additional constraint that it has only H, V, T letters on layer 1 ; on layer 2 put tiles simulating Turing machine computations on T tiles, so that the bottom left corner of each square (a T connected to a V on the left and the H on the bottom) contains the initialization of the Turing computation. This tile set tiles the plane if and only if the machine does not halt starting from the empty string.

References

[1] R. Berger, *The Undecidability of the Domino Problem*, Ph.D. thesis, Harvard University, July 1964.

[2] R. Berger, The undecidability of the domino problem, *Memoirs American Mathematical Society*, **66**(1966).

[3] B. Durand, L. Levin, and A. Shen, Local rules and global order, or aperiodic tilings, *Math. Intelligencer*, **27**(2005), no. 1, 64–68.

[4] B. Durand, A. Romashchenko, and A. Shen, Fixed point and aperiodic tilings, 2007, *preprint.*

[5] B. Grünbaum and G. C. Shephard, *Tilings and patterns*, A Series of Books in the Mathematical Sciences, W. H. Freeman and Company, New York, 1989, an introduction.

[6] A. S. Kahr, E. F. Moore, and H. Wang, Entscheidungsproblem reduced to the $\forall\exists\forall$ case, *Proc. Natl. Acad. Science*, **48**(1962), no. 3, 365–377.

[7] J. Kari, The nilpotency problem of one-dimensional cellular automata, *SIAM J. Comput.*, **21**(1992), no. 3, 571–586.

[8] J. Kari, The tiling problem revisited (extended abstract), in *MCU 2007*, volume 4664 of *LNCS*, pages 72–79, Springer, 2007.

[9] D. Lind and B. Marcus, *An introduction to symbolic dynamics and coding*, Cambridge University Press, Cambridge, 1995.

[10] S. Mozes, Tilings, substitution systems and dynamical systems generated by them, *J. Analyse Math.*, **53**(1989), 139–186.

[11] C. Radin, *Miles of tiles*, volume 1 of *Student Mathematical Library*, American Mathematical Society, Providence, RI, 1999.

[12] R. M. Robinson, Undecidability and nonperiodicity for tilings of the plane, *Inventiones Mathematicae*, **12**(1971), 177–209.

A More pictures!

You will find here some more color pictures. Unfortunately, the construction is very geometrical and colorful. This paper is a compromise between geometrical construction and formal details.

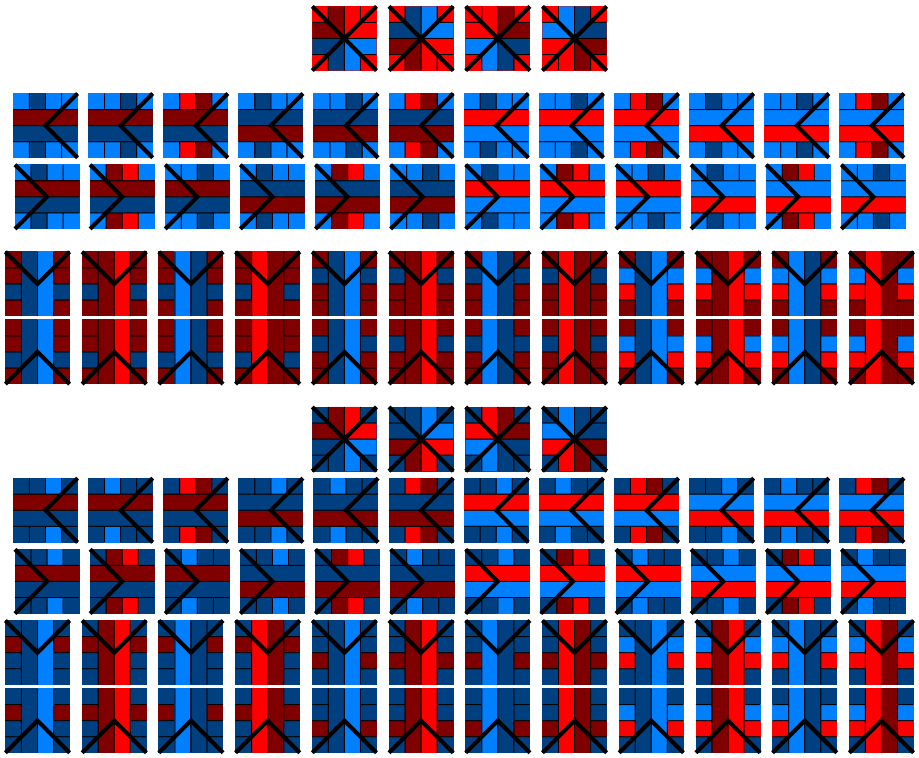


Fig. 5: The 104 tiles of τ

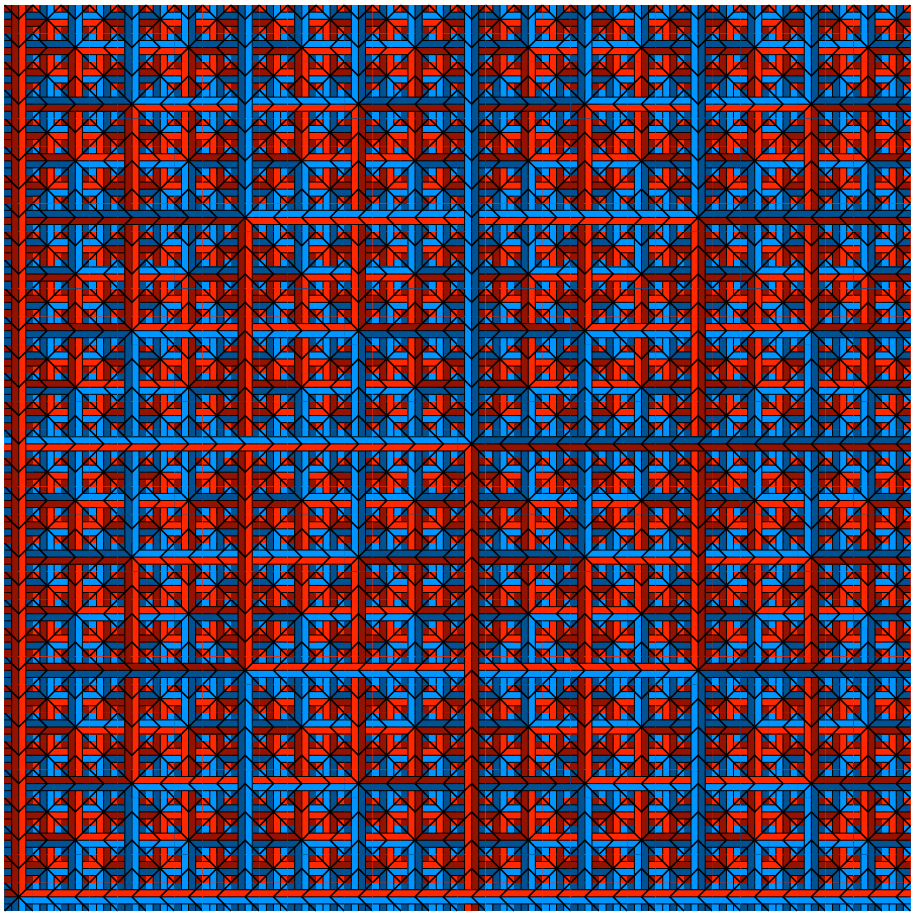


Fig. 6: Part of a tiling by τ

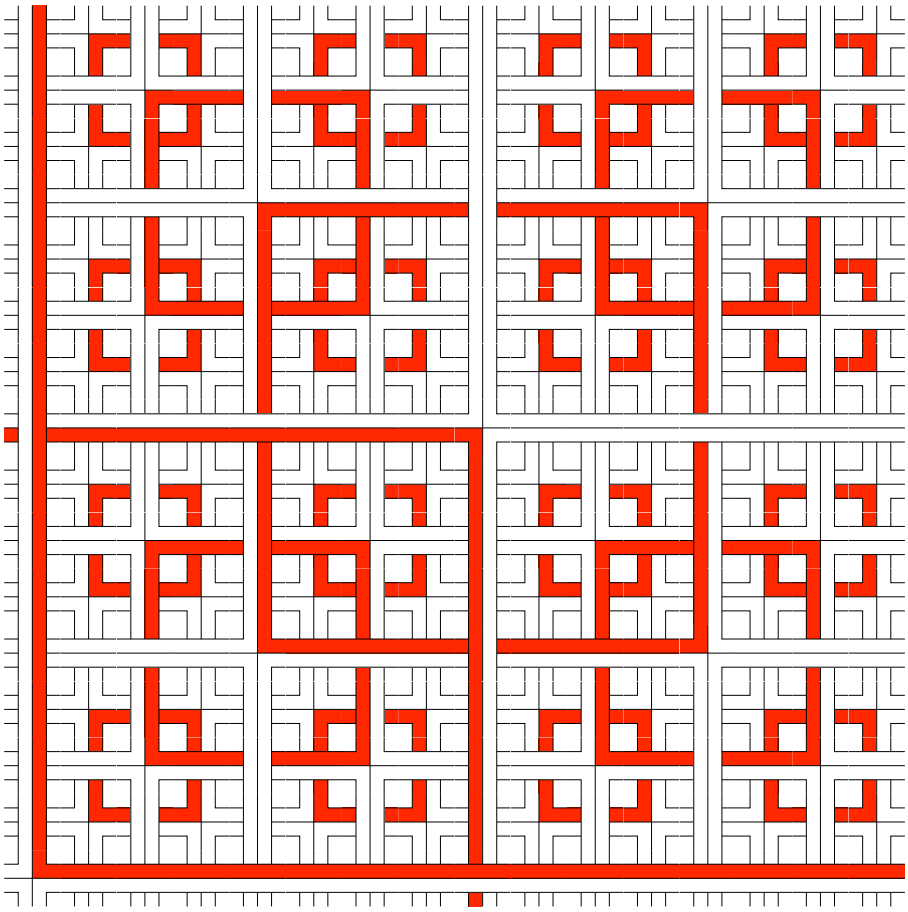


Fig. 7: The $\binom{1}{1}$ squares used to enforce substitutions

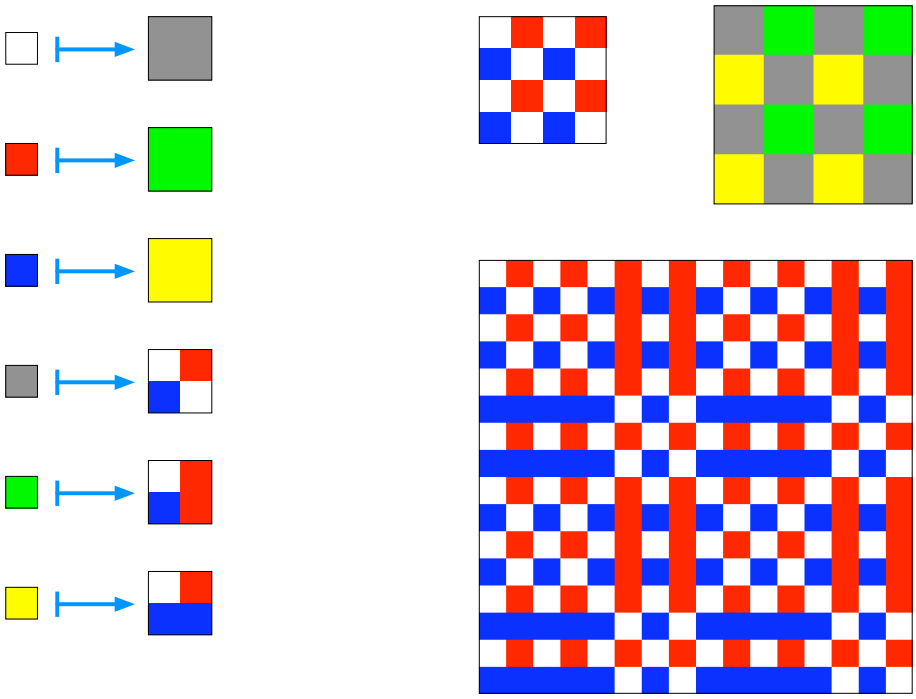


Fig. 8: Graphical $\mathfrak{s}$

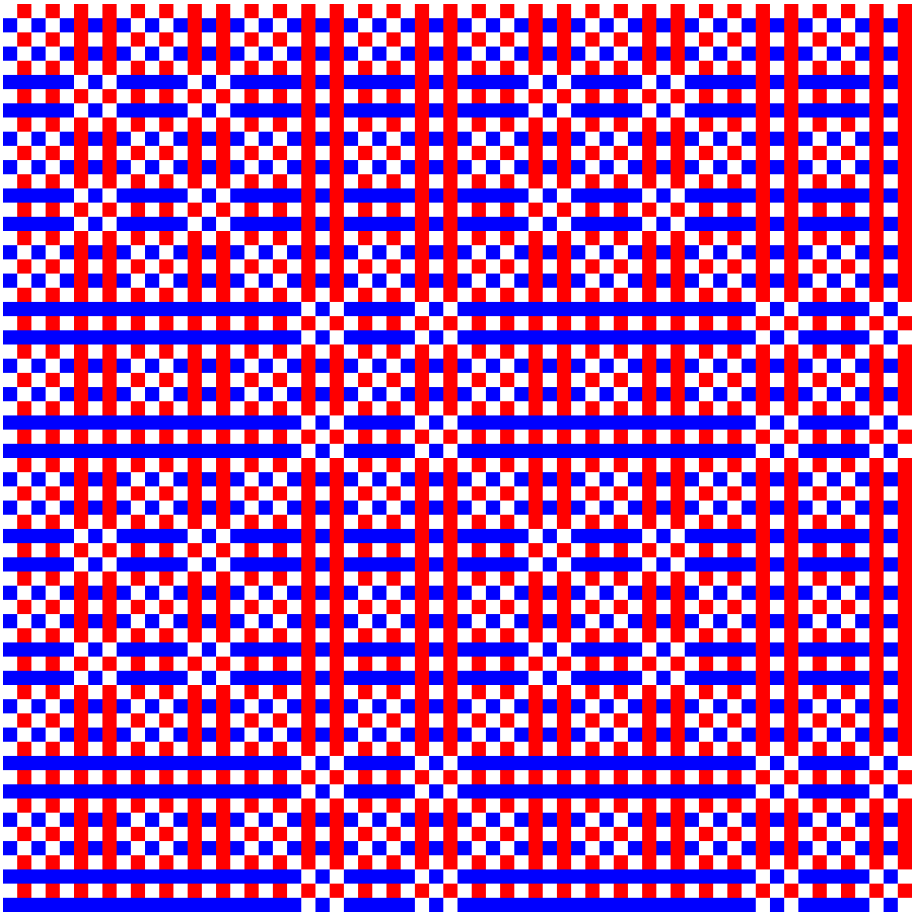


Fig. 9: Detail of Λ_{Θ}

B.2 Tiling the Plane with a Fixed Number of Polyominoes

Résultat récent [U6] montrant l'indécidabilité de la pavabilité du plan par un ensemble de polyominos en nombre borné.

Tiling the Plane with a Fixed Number of Polyominoes

Nicolas Ollinger

Laboratoire d'informatique fondamentale de Marseille (LIF),
Aix-Marseille Université, CNRS,
39 rue Joliot-Curie, 13 013 Marseille, France,
Nicolas.Ollinger@lif.univ-mrs.fr

Abstract. Deciding whether a finite set of polyominoes tiles the plane is undecidable by reduction from the Domino problem. In this paper, we prove that the problem remains undecidable if the set of instances is restricted to sets of 5 polyominoes. In the case of tiling by translations only, we prove that the problem is undecidable for sets of 11 polyominoes.

Introduction

Tiling the plane given a finite set of tiles is an old and fascinating problem. For an survey on tilings, the reader is invited to consult Grünbaum and Shephard [1]. A celebrated computability result by Berger [2] is the undecidability of the Domino problem: given a finite set of Wang tiles, unit squares with colored edges, decide if the Wang tiles can tile the whole plane so that matching edges share a same color. A polyomino is a simple kind of tile: it consists of rookwise connected unit squares. Golomb [3] studied tiling by polyominoes and proved in [4] that the Domino problem can be reduced to deciding if a finite set of polyominoes tiles the plane. The reduction can be achieved by a classical encoding of Wang tiles by polyominoes that preserves tilings. In this reduction, the number of polyominoes is equal to the initial number of Wang tiles. A natural question arises: what happens if we consider the tiling problem for a fixed number of polyominoes? From this previous result, two cases might happen: (1) the problem is undecidable starting from a certain fixed number of polyominoes (2) the problem is decidable for every fixed number of polyominoes but the family of decision procedures is not itself recursive. As case (1) is more likely to happen, the question is to find the frontier between decidability and undecidability. Such a study of decidability questions with respect to a parameter appears for example in the study of semi-Thue systems or for Post correspondence problem (PCP) where it is shown that $PCP(2)$ is decidable and $PCP(7)$ is undecidable [5–7].

Motivated by parallel computing, Wijshoff and van Leeuwen [8] proved that the tilability of the plane by translation of a unique polyomino is decidable. That result was further studied and understood by Beauquier and Nivat [9] who described precisely the tilings by translation generated by a unique polyomino.

This paper is organized as follows. In section 1, we introduce Wang tiles, polyominoes and dented polyominoes, a special variation of polyominoes used in

the constructions. In section 2, we prove that it is undecidable whether a set of 5 polyominoes tiles the plane. In section 3, we deduce from previous section that it is undecidable whether a set of 11 polyominoes tiles the plane by translation. In section 4, we discuss the case of smaller sets of tiles.

1 Definitions

Polyominoes A *polyomino* is a simply connected tile obtained by gluing together rookwise connected unit squares. A *tiling*, of the Euclidian plane, by a set of polyominoes is a partition of the plane such that each element of the partition is the image by an isometry of a polyomino of the set. A *tiling by translation* is a tiling where isometries are restricted to translations. A tiling is *periodic* if it is invariant by translation, *biperiodic* if it is invariant by two non-colinear translations, *aperiodic* if it is not periodic. A set of polyominoes is *aperiodic* if it admits a tiling and all its tilings are aperiodic.

A tiling is *discrete* if all the vertices of the unit squares composing the polyominoes are aligned on the grid $\mathbb{Z}^2$. If a tiling is not discrete, the tiling can be split into two tilings of a half-plane along a line going through an edge along which two unit squares are not aligned. By shifting one half-plane to align the tiles, and iterating this process, one obtains the following lemma.

Lemma 1. *A set of polyominoes admits a tiling if and only if it admits a discrete tiling.*

In this paper where we deal with tilability, we only consider discrete tilings, thanks to this lemma. Thus, a polyomino can be considered as a finite, simply connected, subset of $\mathbb{Z}^2$ and a tiling by a set of polyominoes is a partition of $\mathbb{Z}^2$ where each element is the image by an isometry of an element of the set. Each such isometry can be decomposed into a translation and one out of 8 elementary transformations obtained by composing right angle rotations and mirroring. A sample polyomino and its 8 transformations are represented in Fig. 1.

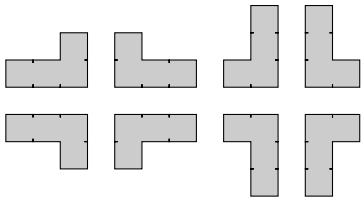
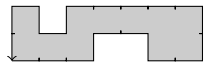


Fig. 1: A polyomino and its 8 transformations

Polyominoes are compactly described by their contour words. A *contour word* of a polyomino is a (finite) word on the alphabet $\{e, w, n, s\}$ describing a walk along the outline of the polyomino starting from and ending to a vertex of the

boundary of the polyomino where e is an east move $(1, 0)$, w is a west move $(-1, 0)$, n is a north move $(0, 1)$ and s is a south move $(0, -1)$. A word is a contour word if and only if the associated path does not cross itself. A polyomino with a pointed contour word is represented in Fig. 2



$e^3ne^2se^2n^2w^5swnws^2$

Fig. 2: A polyomino with a pointed contour word

Wang tiles A *Wang tile* is a unit square with colored edges. A tiling by a set of Wang tiles is a discrete tiling by tiles of the set such that along each edge the colors match on both sides. The *Domino problem* is the following decision problem: given a finite set of Wang tiles, decide whether it admits a tiling.

Theorem 1 (Berger [2]). *The Domino problem is undecidable.*

The *Polyomino problem* is the following decision problem: given a finite set of polyominoes, decide whether it admits a tiling. By a reduction from the Domino problem to the Polyomino problem, Golomb [4] proved the undecidability of the Polyomino problem.

Theorem 2 (Golomb [4]). *The Polyomino problem is undecidable.*

The reduction proceeds as follows. Given a finite set of Wang tiles, Golomb encodes each tile into a big squarish polyomino. Special bumps and dents are added to the corners of the tiles to force both alignment and orientation of the tiles: if one of the encoding polyominoes appears with an orientation, all the other tiles of the tiling have to use the same orientation. Special bumps and dents are used along the sides of the big polyominoes to encode the colors of the Wang tiles. Quotienting the set of tilings of the set of encoding polyominoes by isometries, it is in bijection with the set of tilings of the given set of Wang tiles.

Dented polyominoes A *dented polyomino* is a polyomino with edges labeled by a shape and an orientation. The four possible orientations $\{\mathsf{p}, \mathsf{q}, \mathsf{b}, \mathsf{d}\}$ and their interpretation depending on the direction of the edge are depicted on Tab. 1 for a sample shape. On a contour word, inside shapes define bumps and outside shapes define dents. A tiling by a set of dented polyominoes is a tiling by the corresponding set of polyominoes where bumps and dents match along edges.

Dented polyominoes provide a convenient tool to construct complicated sets of polyominoes. These polyominoes with puzzle bumps and dents can be easily converted into polyominoes.

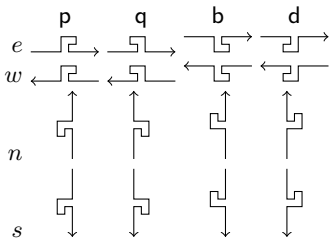


Table 1: Encoding of bumps and dents orientation

Lemma 2. *Every finite set of dented polyominoes can be encoded as a finite set of polyominoes such that their sets of tilings are in one-to-one correspondence.*

Proof. In order to guarantee that bumps and dents do not interfere with the matching conditions of polyominoes, the idea is to rescale the polyominoes. For all $k \in \mathbb{Z}^+$, a k-rescaling of a set of polyominoes consists into scaling the polyominoes by a factor k, *i.e.*, replacing each unit square by a square of k by k unit squares. Tilings are preserved by rescaling: the set of tilings of a set of polyominoes is in one-to-one correspondence with the set of tilings of its k-rescaling.

To encode a finite set of dented polyominoes into a finite set of polyominoes: first, rescale the set of polyominoes by a factor far bigger than the size of any shape of its bumps and dents; then, add bumps and dents in the middle of each rescaled edge. ■

2 Tiling with a fixed number of polyominoes

The k-Polyomino problem is the following decision problem: given a set of k polyominoes, decide whether it admits a tiling. This section is dedicated to the proof of the following theorem.

Theorem 3. *The* 5*-Polyomino problem is undecidable.*

We will proceed by reduction of the Domino problem. Given a finite set τ of Wang tiles, we construct a set of 5 dented polyominoes $P(\tau)$ such that, up to isometry, the set of tilings of τ is in one-to-one correspondence with the set of tilings of $P(\tau)$. The proof goes as follows. First, we describe the construction of $P(\tau)$. Then, we explain how to encode any tiling of τ by a tiling of $P(\tau)$. Finally, we show that any tiling of $P(\tau)$ encodes a tiling of τ.

2.1 Encoding a set of Wang tiles

Let τ be a set of N Wang tiles. The set of dented polyominoes $P(\tau)$ consists of the following 5 tiles, represented in Fig. 3:

meat encodes all tiles of the set τ sequentially;

jaw acts as a selector to select exactly one tile of the *meat*;
filler is used for padding the blank leaved by the *meat* inside the *jaw*;
tooth erases the bits on the *meat* so that it fits inside the *jaw*;
wire links *meat* pieces together to verify tiling constraints.

More formally, the dented polyominoes use 4 different shapes for bumps and dents, detailed on Tab. 2.

	blank	*bit*	*marker*	*inside*
shape				
notation		b	m	i
order	1	4	4	2
bump		*wire*, *tooth*	*meat*, *filler*	*tooth*, *filler*
dent		*meat*	*jaw*	*jaw*

Table 2: Types of bumps and dents

Let k be a large enough integer and choose an encoding on $k-4$ bits of the colors of the set of Wang tiles (horizontal and vertical colors can use different encodings). Let (a^i_j), (b^i_j), (c^i_j), (d^i_j) be respectively the north, east, south, and west binary encoding of the tiles where i is the tile index from 1 to N and j is the bit index from 1 to $k-4$. Let $(\mathfrak{a}^i_j)$ be the encoding of the k bits, by adding prefix 00 and suffix 01, of (a^i_j) on the alphabet $\{\mathsf{b},\mathsf{d}\}$. Let $(\mathfrak{b}^i_j)$ be the encoding of the k bits, by adding prefix 10 and suffix 11, of (b^i_j) on the alphabet $\{\mathsf{p},\mathsf{q}\}$. Let $(\mathfrak{c}^i_j)$ be the encoding of the k bits, by adding prefix 00 and suffix 01, of (c^i_j) on the alphabet $\{\mathsf{b},\mathsf{d}\}$. Let $(\mathfrak{d}^i_j)$ be the encoding of the k bits, by adding prefix 10 and suffix 11, of (d^i_j) on the alphabet $\{\mathsf{p},\mathsf{q}\}$. The dented polyominoes are given by their contour words on Tab. 3.

tooth: $e^{\mathsf{b}}_i n w^{\mathsf{q}}_b s$

wire: $e^{\mathsf{b}}_b n^5 w^{2N(k+1)+1} n^4 w^{\mathsf{p}}_b s^5 e^{2N(k+1)+1} s^4$

filler: $e^{\mathsf{b}}_m \left(s e^{\mathsf{b}}_i\right)^k \left(e^{\mathsf{b}}_i n\right)^k e^{\mathsf{d}}_m n w^{\mathsf{q}}_m \left(n w^{\mathsf{p}}_i\right)^k \left(w^{\mathsf{p}}_i s\right)^k w^{\mathsf{p}}_m s$

jaw: $e^{\mathsf{q}}_m \left(e^{\mathsf{p}}_m \left(n e^{\mathsf{p}}_i\right)^k \left(e^{\mathsf{p}}_i s\right)^k e^{\mathsf{q}}_m\right)^{N-1} s \left(w^{\mathsf{d}}_m \left(s w^{\mathsf{d}}_i\right)^k \left(w^{\mathsf{d}}_i n\right)^k w^{\mathsf{b}}_m\right)^{N-1} w^{\mathsf{d}}_m s^4 e^{2(N-1)(2k+2)+4} n^4$
$w^{\mathsf{b}}_m \left(w^{\mathsf{d}}_m \left(s w^{\mathsf{d}}_i\right)^k \left(w^{\mathsf{d}}_i n\right)^k w^{\mathsf{b}}_m\right)^{N-1} n \left(e^{\mathsf{p}}_m \left(n e^{\mathsf{p}}_i\right)^k \left(e^{\mathsf{p}}_i s\right)^k e^{\mathsf{q}}_m\right)^{N-1} e^{\mathsf{p}}_m n^4 w^{2(N-1)(2k+2)+4} s^4$

meat: $\prod_{i=1}^{N} \left(e^{\mathsf{b}}_m \prod_{j=1}^{k-1} \left(e_b^{\mathfrak{a}^i_j} s\right) e_b^{\mathfrak{a}^i_k} \prod_{j=1}^{k-1} \left(e_b^{\mathfrak{b}^i_j} n\right) e_b^{\mathfrak{b}^i_k} e^{\mathsf{d}}_m\right) n$
$\prod_{i=N}^{1} \left(w^{\mathsf{q}}_m \prod_{j=k}^{2} \left(w_b^{\mathfrak{c}^i_j} n\right) w_b^{\mathfrak{c}^i_1} \prod_{j=k}^{2} \left(w_b^{\mathfrak{d}^i_j} s\right) w_b^{\mathfrak{d}^i_1} w^{\mathsf{p}}_m\right) s$

Table 3: Contour encoding of the tiles

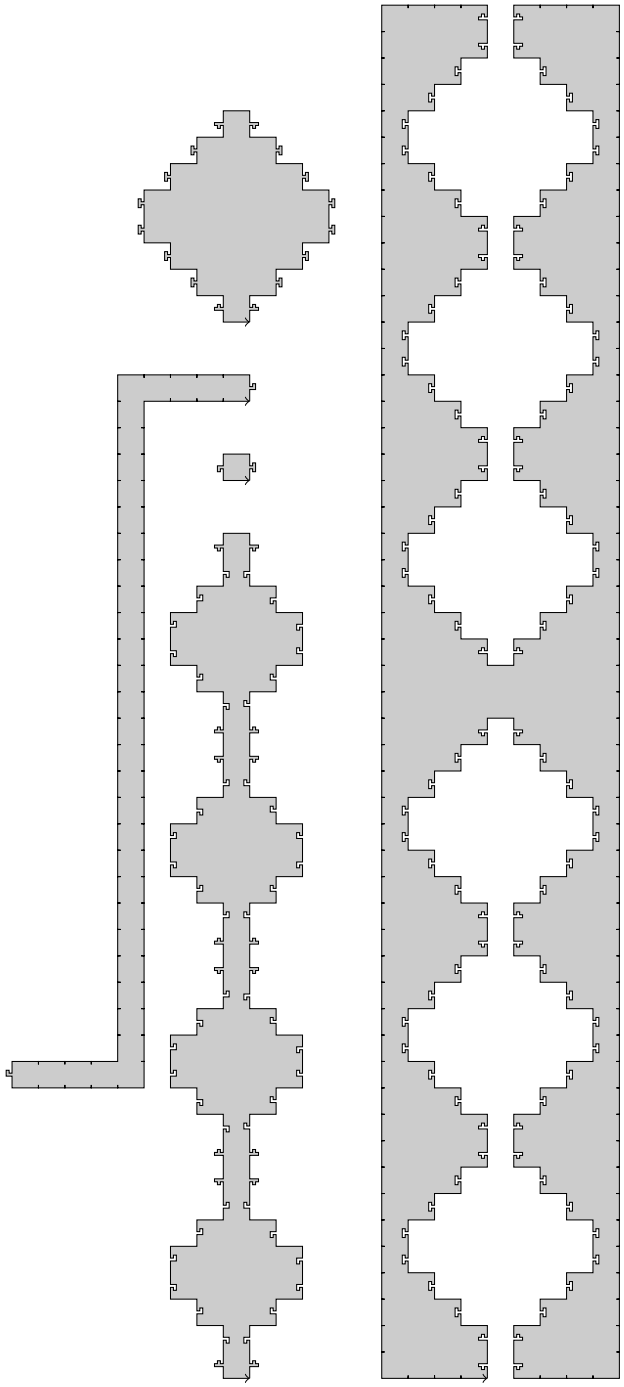


Fig. 3: Tiles *(rotated to fit in page)*
From left to right and bottom to top : *meat*, *tooth*, *wire*, *filler*, *jaw*
Notice that here $N = 4$ and $k = 3$ to fit the page (in the text $k > 4$).

Notice the following important properties of these tiles. The *meat* consists of a sequence of N diamonds decorated with *bit* shapes with a prefix and a suffix *marker* shape pointing to the diamond. If one connects a *tooth* in each *bit* dent of a diamond of the *meat*, the diamond becomes similar to the *filler*. Moreover, both inside parts of a *jaw* consists of $N-1$ places to put a *filler* plus a *marker* shape at the entry of the *jaw*, pointing outside.

2.2 Encoding tilings by Wang tiles

Let us first prove the following lemma.

Lemma 3. *Every tiling by τ can be encoded as a tiling by $P(\tau)$.*

The set of dented polyominoes is designed to encode a Wang tile by selecting one diamond of a *meat*, hiding the other diamonds using two *jaws* on the left and the right, padding inside the *jaws* with *teeth* and *fillers*, as represented in Fig. 4. The colors of the tile are propagated to the four neighbor tiles using *wires*.

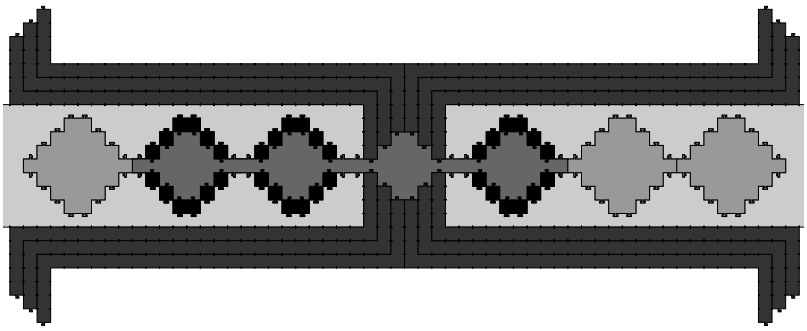


Fig. 4: Encoding of a Wang tile including inter-tiles wires

Let $T \in \tau^{\mathbb{Z}^2}$ be a tiling by τ. Each north-west diagonal ($x+y=i$ for the ith diagonal) is encoded as a line of Wang tile encodings where a *jaw* connects tile $T(x, i-x)$ to tile $T(x+1, i-x-1)$. These lines of encoding are put on top of each other with a slight translation so that tile $T(x, i-x)$ is connected by *wires* to $T(x+1, i-x)$, $T(x, i-x+1)$, $T(x-1, i-x)$ and $T(x, i-x-1)$, as represented in Fig. 5. Notice that the choices made for $\mathfrak{a}$, $\mathfrak{b}$, $\mathfrak{c}$, and $\mathfrak{d}$ permits such a connection only if the Wang tiling is valid. Thus, one obtains a tiling of the dented polyominoes.

2.3 Every tiling encodes a tiling

Now, we prove the following lemma.

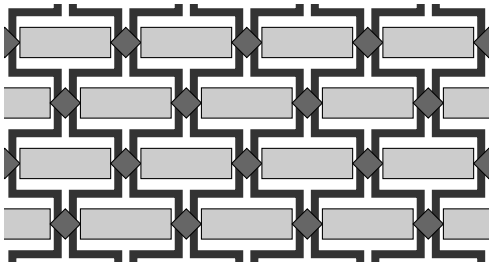


Fig. 5: Wiring of Wang tiles

Lemma 4. *Every tiling by $P(\tau)$ encodes a tiling by τ.*

Consider a tiling by the dented polyominoes. We first show that it must contain a *jaw*. Consider any tile of the tiling. If it is a *jaw*, we are done. Examine Tab. 2. If it is a *meat* or a *filler*, it has a *marker* bump that should be linked to a dent only found on a *jaw*. If it is a *tooth*, it has an *inside* bump that should be linked to a dent only found on a *jaw*. Finally, if it is a *wire*, it has *bit* bump that should be linked to a *meat*, itself connected to a *jaw*.

Consider a *jaw* tile of the tiling. To fill all the *marker* bumps, only *filler* and *meat* tiles can be used. As *fillers* have *inside* bumps, they can only be used completely inside the *jaw*. Thus, the *markers* on the extremities of the *jaw* have to be filled by the dents of a *meat*. Consider the *meat* that fills a *marker* at the extremity of the *jaw*. As *marker* bumps only appear inside *jaws*, the *marker* dent on the other side of the diamond of the *meat* next to the *jaw* has to be at the extremity of a next *jaw*. The only possibility to fill the gap in between the *jaw* and the *meat* locking its extremity is to use *fillers* and *teeth*. By now, we have proved that each *jaw* of the tiling appears in a biinfinite line (or column if rotated) of alternating *jaws* and *meats* where each *meat* has exactly one selected diamond outside the *jaws*.

Consider now the diamond of each *meat* appearing outside the *jaws*: it has *bit* bumps. A *bit* dent is found only on *teeth* and *wires*. As a *tooth* cannot appear outside a *jaw* (it as an *inside* dent), only *wires* can be connected to these bumps. Consider the two *bit* bumps side by side at the top of the diamond hill. The only possibility for two *wires* to appear side by side is to have the left one point to the left and the right one point to the right. This enforces all the *wires* in between the *jaw* and the top hill *wire* to point in the same direction: left ones to the left and right ones to the right. Thus, every tiling by dented polyominoes consists of biinfinite lines (or columns) of selected *meat* diamonds connected by *wires* in a lattice way as on Fig. 5.

Due to isometries, it remains to prove that all the selected diamonds have the same orientation. This part is enforced by the prefix/suffix trick in the $\mathfrak{a}$, $\mathfrak{b}$, $\mathfrak{c}$, and $\mathfrak{d}$ encoding: the only possibility for a diamond to be connected to another diamond is that prefix and suffix match, thus both should be oriented in the same way. Thus, the tiling is the image by an isometry of a tiling by Wang tiles.

We have proved that every tiling by $P(\tau)$ encodes a tiling by τ, achieving the reduction.

3 Tiling by translation

The k-Polyomino translation problem is the following decision problem: given a set of k polyominoes, decide if it admits a tiling by translation. Using the result of previous section, one obtains the following.

Theorem 4. *The* 11*-Polyomino translation problem is undecidable.*

To prove this corollary, for any finite set of Wang tiles τ, construct a set of 11 polyominoes as follows. Consider the set of 5 dented polyominoes $P(\tau)$. To encode any tiling by τ into a tiling by $P(\tau)$ as done in the previous section, we use exactly:

1 transformation of *meat*;
1 transformation of *jaw*;
1 transformation of *filler*;
4 transformations of *wire*;
4 transformations of *tooth*.

The set of 11 polyominoes tiling by translation consists exactly of these tile transformations. These dented polyominoes admit a tiling if and only if the set of Wang tiles admits a tiling.

4 Going further

What can be said about tilability for sets of less that 5 polyominoes? or less than 11 polyominoes for tilings by translation? In the case of 1 polyomino, it is decidable for tiling by translation and still open for tiling with isometries.

Theorem 5 (Wijshoff and van Leeuwen [8], Gambini and Vuillon [10]). *The* 1*-Polyomino translation problem is decidable in time quadratic in the size of the contour word.*

Open Problem 1. *Is the* 1*-Polyomino problem decidable?*

To prove the undecidability of the Polyomino problem, one has to be able to construct aperiodic sets of polyominoes. Ammann *et al* provide a set of 2 polygonal tiles with colors and 3 polygonal tiles with bumps and dents that admits only aperiodic tilings. This set is convertible into polyominoes.

Theorem 6 (Ammann *et al* [11]). *There exists an aperiodic set of* 3 *polyominoes.*

Theorem 7 (Ammann *et al* [11]). *There exists an aperiodic set of* 8 *polyominoes for tiling by translation.*

Open Problem 2. *Is the* k*-Polyomino problem decidable for* $3 \leqslant k < 5$*?*

Open Problem 3. *Is the* k*-Polyomino translation problem decidable for* $8 \leqslant k < 11$*?*

Acknowledgement

The author thanks Bruno Durand for challenging him with the decision problem of tiling the plane with a fixed number of polyominoes.

References

1. Grünbaum, B., Shephard, G.C.: Tilings and patterns. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York (1989)
2. Berger, R.: The undecidability of the domino problem. Memoirs American Mathematical Society **66** (1966)
3. Golomb, S.W.: Tiling with polyominoes. Journal of Combinatorial Theory **1**(2) (1966) 280–296
4. Golomb, S.W.: Tiling with sets of polyominoes. Journal of Combinatorial Theory **9**(1) (1970) 60–71
5. Post, E.L.: A variant of a recursively unsolvable problem. Bulletin of the American Mathematical Society **52** (1946) 264–268
6. Claus, V.: Some remarks on PCP(k) and related problems. Bulletin of the EATCS **12** (1980) 54–61
7. Matiyasevich, Y., Sénizergues, G.: Decision problems for semi-thue systems with a few rules. Theoretical Computer Science **330**(1) (2005) 145–169
8. Wijshoff, H.A.G., van Leeuwen, J.: Arbitrary versus periodic storage schemes and tessellations of the plane using one type of polyomino. Information and Control **62**(1) (1984) 1–25
9. Beauquier, D., Nivat, M.: On translating one polyomino to tile the plane. Discrete and Computational Geometry **6**(1) (1991) 575–592
10. Gambini, I., Vuillon, L.: An algorithm for deciding if a polyomino tiles the plane. Theoretical Informatics and Applications **41**(2) (2007) 147–155
11. Ammann, R., Grünbaum, B., Shephard, G.: Aperiodic tiles. Discrete and Computational Geometry **8**(1) (1992) 1–25

B.3 Periodicity and Immortality in Reversible Computing

Version courante de [U5], un article écrit en collaboration avec J. Kari consacré à l'étude de l'indécidabilité de la périodicité et de l'immortalité de trois modèles de calcul réversibles : automates à compteurs, machines de Turing et automates cellulaires. Une première version [C6], plus courte, a été présentée à MFCS'2008.

Periodicity and Immortality in Reversible Computing ⋆,⋆⋆

J. Kari [a] and N. Ollinger [b,1]

[a] *Department of Mathematics, FIN-20014 University of Turku, Finland*

[b] *Laboratoire d'informatique fondamentale de Marseille (LIF), Aix-Marseille Université, CNRS, 39 rue Joliot-Curie, 13 013 Marseille, France*

Abstract

We investigate the decidability of the periodicity and the immortality problems in three models of reversible computation: reversible counter machines (RCM), reversible Turing machines (RTM) and reversible one-dimensional cellular automata (RCA). We discuss programming techniques for the first two models. Immortality and periodicity are properties that describe the behavior of the model starting from arbitrary initial configurations: immortality is the property of having at least one non-halting orbit, while periodicity is the property of always eventually returning back to the starting configuration. It turns out that periodicity and immortality problems are both undecidable in all three models. We also show that it is undecidable whether a (not-necessarily reversible) Turing machine with moving tape has a periodic orbit.

Key words: reversibility, counter machines, turing machines, cellular automata

Introduction

Reversible computing is the classical counterpart of quantum computing. Reversibility refers to the fact that there is an inverse process to retrace the computation back in time, i.e., the system is time invertible and no information is ever lost. Much of the research on reversible computation is motivated

⋆ Work supported by grants of the French ANR and Academy of Finland # 211967

⋆⋆A preliminary version of this work, with sketched proofs, first appeared in [7]

[1] Corresponding author (Nicolas.Ollinger@lif.univ-mrs.fr)

by the Landauer's principle which states a strict lower bound on the amount of energy dissipation which must take place for each bit of information that is erased [9]. Reversible computation can, in principle, avoid this generation of heat.

Reversible Turing machine (RTM) was the earliest proposed reversible computation model [10,1]. Since then, reversibility has been investigated within other common computation models such as Minsky's counter machines [11,12] and cellular automata [5]. In particular, reversible cellular automata (RCA) have been extensively studied due to the other physics-like attributes of cellular automata such as locality, parallelism and uniformity in space and time of the update rule.

All three reversible computation models are Turing complete: they admit simulations of universal Turing machines, which naturally leads to various undecidability results for reachability problems. In this work we view the systems, however, rather differently by investigating their behavior from arbitrary starting configurations. This is more a dynamical systems approach. Each device is understood as a transformation $F : X \longrightarrow X$ acting on its configuration space X. In all cases studied here (counter machines, two Turing machine models – with moving head and with moving tape – and cellular automata) space X is endowed a topology under which F is continuous. In the cases of Turing machines with moving tape and cellular automata, it is the compact and metrizable topology obtained as the enumerable infinite product of the discrete topology on each finite component of a configuration. The action F may be partial, so that it is undefined for some elements of X. Configurations on which F is undefined are called *halting*. We call F *immortal* if there exists a configuration $x \in X$ that never evolves into a halting configuration, that is, $F^n(x)$ is defined for all positive integers n. In contrast, a *mortal* system eventually halts, regardless of the starting configuration. We call F *uniformly mortal* if a uniform time bound n exists such that $F^n(x)$ is not defined for any $x \in X$. If F is continuous, X compact, and the set of halting configurations open then mortality and uniform mortality are equivalent concepts. This means that mortal Turing machines and cellular automata are automatically uniformly mortal. In contrast, a counter machine may be mortal without being uniformly mortal. (A simple example is a one-counter machine where the counter value is repeatedly decremented until it becomes zero and the machine halts.)

Periodicity, on the other hand, is defined for *complete* systems: systems without halting configurations. We call total $F : X \longrightarrow X$ *uniformly periodic* if there is a positive integer n such that F^n is the identity map. *Periodicity* refers to the property that every configuration is periodic, that is, for every $x \in X$ there exists time n such that $F^n(x) = x$. Periodicity and uniform periodicity are equivalent concepts in the cases of cellular automata (Section 3.3) and

Turing machines under both modes (Section 2.1), while a counter machine can be periodic without being uniformly periodic (Example 2 in Section 1.1). It is also the case that a Turing machine is periodic under the moving tape mode if and only if it is periodic under the moving head mode (Section 2.1).

In this work we are mainly concerned with decidability of these concepts. Immortality of unrestricted (that is, not necessarily reversible) Turing machines was proved undecidable already in 1966 by Hooper [4]. The proof of our main result (Theorem 20) is a reversible variant of Hooper's approach where infinite searches during counter machine simulations by a Turing machine are replaced by recursive calls to the counter machine simulation itself with empty initial counters. Using reversible counter machines, the recursive calls can be unwound once the search is complete. In a sense this leads to a simpler construction than in Hooper's original article.

Our result also answers an open problem of control theory from [3]. That paper pointed out that if the immortality problem for reversible Turing machines is undecidable, then so is observability for continuous rational piecewise-affine planar homeomorphisms.

As another corollary we obtain the undecidability of the periodicity of Turing machines (Theorem 23). The related problem of determining if a given Turing machine has at least one periodic orbit (under the moving tape mode) is proved undecidable for reversible, non-complete Turing machines, and for non-reversible, complete Turing machines. The problem remains open under reversible and complete machines. The existence of periodic orbits in Turing machines and counter machines have been investigated before in [8,2]. Article [8] formulated a conjecture that every complete Turing machine (under the moving tape mode) has at least one periodic orbit, while [2] refuted the conjecture by providing an explicit counter example. The counter example followed the general idea of [4] in that recursive calls were used to prevent unbounded searches. In [2] it was also shown that it is undecidable if a given complete counter machine has a periodic orbit. We show that this is the case even under the additional constraint of reversibility (Theorem 10).

In Theorem 28 we reduce the periodicity problem of reversible Turing machine into the periodicity problem of one-dimensional cellular automata. The immortality problem of reversible cellular automata has been proved undecidable in [6]. Our proofs for the undecidability of immortality (Theorem 3) and periodicity (Theorem 6) among reversible counter machines follow the techniques of [12]. Interestingly, the uniform variants of both immortality and periodicity problems are decidable for counter machines (Theorems 5 and 8).

The paper is organized into three parts dealing with RCM (Section 1), with RTM (Section 2) and with RCA (Section 3). Each part consists of five sub-

sections on (1) definitions, (2) programming, (3) the immortality problem, (4) the periodicity problem, and (5) the existence of periodic orbits.

1 Reversible Counter Machines

1.1 Definitions

Following [12], we define special counter machine instructions for a simpler syntactic characterization of local reversibility and forget about initial and accepting states as we are only interested in dynamical properties.

Let $\Upsilon = \{Z, P\}$ be the set of test values and $\Phi = \{-, 0, +\}$ be the set of counter operations. A *k-counter machine* M is a triple (S, k, T) where S is a finite set of states, $k \in \mathbb{N}$ is the number of counters, and $T \subseteq S \times \mathbb{Z}_k \times (\Upsilon \cup \Phi) \times S$ is the transition table of the machine. Quadruple $(s, i, \phi, t) \in T$ is called a test instruction if $\phi \in \Upsilon$ and a modifying instruction if $\phi \in \Phi$.

A configuration $\mathfrak{c}$ of the machine is a pair (s, v) where $s \in S$ is a state and $v \in \mathbb{N}^k$ is the value of the counters. The machine can transform a configuration $\mathfrak{c}$ into a configuration $\mathfrak{c}'$ in one step, noted as $\mathfrak{c} \vdash \mathfrak{c}'$, by applying an instruction $\iota \in T$. Test instructions (s, i, Z, t) and (s, i, P, t) can be applied to configurations (s, v) where $v(i) = 0$ and $v(i) > 0$, respectively, leading to the configuration (t, v). Modifying instructions $(s, i, -, t), (s, i, 0, t)$ and $(s, i, +, t)$ transform configuration (s, v) into (t, v') where $v'(i) = v(i) - 1$, $v'(i) = v(i)$ and $v'(i) = v(i) + 1$, respectively, and $v'(j) = v(j)$ for all $j \neq i$. However, if $v(i) = 0$ then $(s, i, -, t)$ cannot be applied on configuration (s, v), as all counters must remain non-negative. The transitive closure of $\vdash$ is noted as $\vdash^*$.

If no instruction can be applied on a configuration then that configuration is *halting*. A counter machine (S, k, T) is *complete* if it has no halting configurations. It is easy to see that (S, k, T) is complete if and only if for every $s \in S$ there exist some $i \in \mathbb{Z}_k$ and states $r, t \in S$ such that one of the following holds:

(i) $(s, i, +, r) \in T$,
(ii) $(s, i, 0, r) \in T$,
(iii) $(s, i, -, r) \in T$ and $(s, i, Z, t) \in T$, or
(iv) $(s, i, P, r) \in T$ and $(s, i, Z, t) \in T$.

A counter machine (S, k, T) is a *deterministic k-counter machine (k-DCM)* if at most one instruction can be applied from any configuration. This is the case if and only if for every $s \in S$ there are at most two instructions

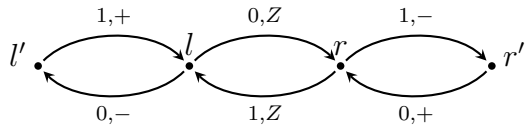


Fig. 1. a complete RCM

$(s, \dots) \in T$, and if there are two such instructions then they must be $(s, i, -, r)$ and (s, i, Z, t), or (s, i, P, r) and (s, i, Z, t) for some $i \in \mathbb{Z}_k$ and $r, t \in S$.

The *transition function* of a deterministic counter machine is the partial function $G : S \times \mathbb{N}^k \to S \times \mathbb{N}^k$ which maps a configuration $\mathfrak{c}$ to the unique configuration $\mathfrak{c}'$ such that $\mathfrak{c} \vdash \mathfrak{c}'$. If $\mathfrak{c}$ is halting then $G(\mathfrak{c})$ is undefined. The transition function is then everywhere defined if and only if the DCM is complete.

The reverse instruction of (s, i, ϕ, t) is the instruction (t, i, ϕ^{-1}, s), where

$$-^{-1} = +,\ 0^{-1} = 0,\ +^{-1} = -,\ Z^{-1} = Z \ \text{ and } \ P^{-1} = P.$$

The reverse T^{-1} of a transition table T is defined as $T^{-1} = \{\iota^{-1} \mid \iota \in T\}$. The *reverse* of counter machine $M = (S, k, T)$ is the machine $M^{-1} = (S, k, T^{-1})$. Note that $\mathfrak{c} \vdash \mathfrak{c}'$ in M if and only if $\mathfrak{c}' \vdash \mathfrak{c}$ in M^{-1}. A *reversible k-counter machine (k-RCM)* is a deterministic k-counter machine whose reverse is deterministic. Clearly a deterministic counter machine is reversible if and only if its partial transition function is one-to-one.

Observe that a complete RCM is not necessarily surjective, that is, its reverse is not necessarily complete. For example, a one-counter machine that repeatedly increments its only counter is reversible and complete, but its reverse has a halting configuration with counter value zero. However, a periodic DCM is surjective:

Lemma 1 *A periodic DCM is reversible with a complete reverse.*

PROOF. The preimage of a configuration $\mathfrak{c}$ with period t always exists as a periodic DCM is complete and is uniquely defined by $G^{t-1}(\mathfrak{c})$. ■

Example 2 *The complete 2-RCM $(\{l, l', r, r'\}, 2, T)$ with the following T is periodic but not uniformly periodic:*

$$T = \{(l, 0, Z, r), (r, 1, Z, l), (l, 0, -, l'), (r, 1, -, r'), (l', 1, +, l), (r', 0, +, r)\}$$

In l, l' tokens are moved from the first counter to the second, and in states r, r' back to the first counter. Its reverse is obtained by swapping $l \leftrightarrow r$ and $l' \leftrightarrow r'$. The machine is depicted on Fig. 1. ◇

1.2 Programming RCM

Let us first recall some classical counter machine programming techniques. After each technique we state a proposition that captures the mortality and periodicity preservation features of the method that we need in the present work.

1.2.1 Stack emulation [11,12]

A stack over an alphabet of arity n can be encoded on a pair of counters: one counter coding the stack itself and one scratch counter which is assumed to contain value 0 before and after each instruction emulation. The stack is simply encoded as the digits in base n of an integer. Empty stack test is emulated by equality to zero test on coding stack. To push letter i, replace value N on coding stack by $nN+i$. To pop a letter, replace value N on coding stack by $\lfloor N/n \rfloor$. Top letter on the stack is $N \mod n$ where N is the value on coding stack. Emulation is operated by transferring value from coding stack counter to scratch counter, then back, computing value modulo n using states. The emulation can be performed in a way that preserves both mortality and reversibility.

As every (reversible) Turing machine is emulated by a (reversible) finite automaton with two stacks, every (reversible) Turing machine can be emulated by a 3-(R)CM (see [11] for a discussion on counter machine universality).

Proposition 1 *One can effectively construct, for a given DTM M, a 3-DCM M' such that M' is mortal if and only if M is mortal. If M is reversible then M' is reversible.*

1.2.2 Counter emulation [11,12]

One can encode k counters on a pair of counters: one counter coding the counters themselves and one scratch counter which is assumed to contain value 0 before and after each instruction emulation. The counters are simply encoded as a product of powers of co-prime numbers $2^\alpha 3^\beta 5^\gamma \dots$ As for stack emulation, each instruction is emulated using products, divisions and modulo computing using the scratch counter. The emulation can be performed in a way that preserves both mortality and reversibility. Also the mortality of the reverse is preserved.

Thus, every (reversible) counter machine can be emulated by a 2-(R)CM.

Proposition 2 *One can effectively construct, for a given k-DCM M, a 2-*

DCM M' such that

- *M' is mortal if and only if M is mortal,*
- *M'^{-1} is mortal if and only if M^{-1} is mortal, and*
- *M' has a periodic orbit if and only if M has a periodic orbit.*

If M is reversible then M' is reversible.

1.2.3 Bookkeeping [12]

A k-DCM can be emulated by a $(k+2)$-RCM using two counters to emulate a bookkeeping stack. Each instruction of the machine is emulated in two steps: first the instruction itself is performed modifying counters as expected, then a finite description of the instruction is stored on the bookkeeping stack. By carefully fixing details, this scheme produces a reversible counter machine, and it preserves mortality.

Thus, every deterministic counter machine can be emulated by a 2-RCM.

Proposition 3 *One can effectively construct, for a given k-DCM M, a $(k+2)$-RCM M' such that M' is mortal if and only if M is mortal.*

Combining all three techniques above shows that every Turing machine can be emulated by a 2-RCM.

1.2.4 Bounded past

A k-RCM can be emulated by a $(k+1)$-RCM with mortal reverse and without periodic orbit: the new counter is simply incremented after each emulated instruction. Growth of the new counter ensures there is no periodic orbit. The reverse, being forced to decrement a bounded counter every two instructions, is mortal. The emulation preserves mortality.

The counter emulation technique above preserves the mortality of the reverse CM, so the number of counters can be reduced from $(k+2)$ to 2 using Proposition 2

Proposition 4 *One can effectively construct, for a given k-RCM M, a 2-RCM M' with mortal reverse, such that M' is mortal if and only if M is mortal.*

1.2.5 Checkpoint [2]

A k-DCM can be emulated by a $(k+2)$-DCM that eventually enters a checkpoint state in bounded time: one counter stores the remaining fuel, the other one stores the burned fuel. One cell of fuel is burned after each emulated instruction. When there is no more fuel, the checkpoint state is entered. The emulation preserves reversibility. A typical use of checkpoint is to use checkpoint state to transform burned fuel back into fuel, incrementing or multiplying it, and to relaunch computation from a fixed state and counter values (typically initial state and empty counters). This ensures that every infinite orbit of the new machine goes through unbounded non-halting segments of the orbit of the initial configuration: the constructed machine is immortal if and only if the initial machine never halts starting from the initial configuration. Adding some bookkeeping, the construction preserves reversibility.

1.3 Undecidability of the Immortality Problem

Hooper's article [4] contains a comment that the immortality problem of Turing machines can be reduced to the immortality problem of counter machines. This we can do, indeed, using Proposition 1. Then, using Propositions 2 and 3, we can obtain the result for 2-RCM. But there is a simple direct argument for the undecidability of counter machine immortality, based on the checkpoint technique from [2].

Theorem 3 *It is undecidable whether a given* 2*-RCM is immortal.*

PROOF. For a given 2-DCM $M = (S, 2, T)$ and initial state $s_0 \in S$ one effectively constructs the following 4-DCM M' that is mortal if and only if M halts from the initial configuration $(s_0, 0, 0)$.

Machine M' performs bounded simulations of M using 4 counters: Counters 1 and 2 represent the two counters of M. Counters 3 and 4 bound the length of the simulation of M: tokens are moved from counter 3 to counter 4 and back. Simulation of M is restarted from the initial configuration $(s_0, 0, 0)$ whenever counter 3 becomes empty. The length of the simulation cycle is $n_3 + n_4$, i.e. the total number of tokens in counters 3 and 4. Whenever the simulation is restarted a new token is added to counter 4, so that the length of the simulation cycle keeps increasing. It is clear that M' has an immortal configuration if and only if configuration $(s_0, 0, 0)$ is immortal in M.

Due to Minsky's classical result on counter machines [11] the halting problem of 2-DCM from initial configuration $(s_0, 0, 0)$ is known to be undecidable — hence we conclude that it is undecidable whether a given 4-DCM is immortal.

Using the bookkeeping and counter emulation techniques of [12] (Propositions 2 and 3) the counter machine can effectively be made reversible and the number of counters can effectively be reduced to two without changing the mortality status of the machine. Hence Theorem 3 follows. ■

Remark 4 *The 2-RCM constructed in the proof can be forced to have mortal reverse using the bounded past technique (Proposition 4).*

It is interesting to note that, in contrast to mortality, it is decidable whether all configurations are mortal within a uniform time bound:

Theorem 5 *It is decidable whether a given k-DCM is uniformly mortal.*

PROOF. Induction on k: The claim is trivial for $k = 0$. For the inductive step, let M be a k-DCM, $k \geq 1$. For $i = 1, 2, \ldots, k$ set counter i to be always positive and test whether the so obtained $(k-1)$-DCM M_i is uniformly mortal. If all k recursive calls return a positive answer, set n to be a common uniform mortality time bound for all k machines M_i. Since counters can be decremented by one at most, we know that configurations of M with some counter value $\geq n$ are mortal. Immortality hence occurs only if there is a period within the finite number of configurations with all counters $< n$. ■

1.4 Undecidability of the Periodicity Problem

Theorem 6 *It is undecidable whether a given complete 2-RCM is periodic.*

PROOF. Let $M = (S, 2, T)$ be a given 2-RCM whose reverse is mortal. In particular, there are no periodic configurations in M. According to Remark 4, it is enough to effectively construct a complete 2-RCM M' that is periodic if and only if M is mortal. Machine M' has state set $S \times \{+, -\}$ where states $(s, +)$ and $(s, -)$ represent M in state s running forwards or backwards in time, respectively. In a halting configuration the direction is switched.

More precisely, $M' = (S \times \{+, -\}, 2, T')$ where T' contains the instruction $[(s, +), i, \phi, (t, +)]$ for each $(s, i, \phi, t) \in T$, and the instruction $[(s, -), i, \phi, (t, -)]$ for each $(s, i, \phi, t) \in T^{-1}$. In addition,

- if for $s \in S$ there is no instruction $(s, \ldots)$ in T then we add to T' the instruction $[(s, +), 0, 0, (s, -)]$,
- if for $s \in S$ there is no instruction $(s, \ldots)$ in T^{-1} then we add to T' the instruction $[(s, -), 0, 0, (s, +)]$,

- if for $s \in S$ the only instruction $(s, \dots)$ in T is (s, i, Z, t) for some $i \in \mathbb{Z}_k$ and $t \in S$ then we add to T' the instruction $[(s,+), i, P, (s,-)]$,
- if for $s \in S$ the only instruction $(s, \dots)$ in T^{-1} is (s, i, Z, t) for some $i \in \mathbb{Z}_k$ and $t \in S$ then we add to T' the instruction $[(s,-), i, P, (s,+)]$,
- if for $s \in S$ the only instruction $(s, \dots)$ in T is (s, i, P, t) or $(s, i, -, t)$ for some $i \in \mathbb{Z}_k$ and $t \in S$ then we add to T' the instruction $[(s,+), i, Z, (s,-)]$,
- if for $s \in S$ the only instruction $(s, \dots)$ in T^{-1} is (s, i, P, t) or $(s, i, -, t)$ for some $i \in \mathbb{Z}_k$ and $t \in S$ then we add to T' the instruction $[(s,-), i, Z, (s,+)]$.

With these instructions,

$$((s,+),u) \vdash ((t,+),v) \text{ in } M' \iff (s,u) \vdash (t,v) \text{ in } M,$$

$$((s,-),u) \vdash ((t,-),v) \text{ in } M' \iff (s,u) \vdash (t,v) \text{ in } M^{-1},$$

$$((s,+),u) \vdash ((t,-),v) \text{ in } M' \iff (s,u) \text{ is halting in } M, s = t \text{ and } u = v, \text{ and}$$

$$((s,-),u) \vdash ((t,+),v) \text{ in } M' \iff (s,u) \text{ is halting in } M^{-1}, s = t \text{ and } u = v.$$

The constructed counter machine M' is reversible and complete. If M is not mortal then it has a non-periodic, non-mortal configuration (s,u), so the configuration $((s,+),u)$ is non-periodic in M'. Conversely, if M is mortal then for every configuration (s,u) in M we have a derivation $(s,u) \vdash^* (t,v)$ where (t,v) is halting in M, and because M^{-1} is mortal we have in M^{-1} a derivation $(s,u) \vdash^* (r,x)$ where (r,x) is halting in M^{-1}. This means that in M' we have

$$\begin{aligned}((s,+),u) \vdash^* ((t,+),v) \vdash ((t,-),v) \vdash^* \\ \vdash^* ((s,-),u) \vdash^* ((r,-),x) \vdash ((r,+),x) \vdash^* ((s,+),u),\end{aligned}$$

so $((s,+),u)$ and $((s,-),u)$ are periodic. ■

Example 7 *The 2-RCM $(\{s,t\},2,T)$ with $T = \{(s,1,-,t),(t,0,+,s)\}$ is mortal and its reverse is also mortal. The machine moves tokens from counter 1 to counter 0 until counter 1 becomes empty. The construction in the proof of Theorem 6 results in the periodic 2-RCM of Example 2, where states $(s,+)$, $(t,+)$, $(s,-)$ and $(t,-)$ are called r, r', l and l', respectively.* ◇

Analogously to Theorem 5 one can prove the following result.

Theorem 8 *It is decidable whether a given k-DCM is uniformly periodic.*

PROOF. Let M be a given k-DCM. We prove using induction on k that an algorithm exists that tests M for uniform periodicity and that returns a period.

1° If $k = 0$ then the configuration space if finite and can be effectively checked.

2° (Inductive step) Let $k > 0$ and suppose that an algorithm exists to check the uniform periodicity of given $(k-1)$-DCM. For every $i = 1, 2, \ldots k$, construct the $(k-1)$-DCM M_i obtained from M by removing counter i and pretending in all transitions that counter i has always positive value. This corresponds to the idea of having a very large value in counter i. Clearly, if M is uniformly periodic then M_i are uniformly periodic. So we recursively check if all M_i are uniformly periodic. If any one is not, we can conclude that M is not uniformly periodic either.

Suppose then that each M_i is uniformly periodic with period p_i. Note that this does not yet imply that all configurations of M with large value in counter i are periodic – it can namely happen that the value in counter i gets incremented or decremented in otherwise periodic orbits. To remove that possibility we set counter i to value p_i, set other $k-1$ counters to values $\leq p_i$ (and repeat the test for all combinations of such values in all counters) and test whether after p_i iterations of M counter i has returned to value p_i. If not, we know that M is not uniformly periodic.

Next we set $p = \mathrm{lcm}(p_1, p_2, \ldots, p_k)$ so that we know that all configurations where some counter has value $\geq p$ are periodic with period p. All that remains to be done is to check for every configuration c with all counters $< p$ that either c is periodic with period p or that c is periodic in such a way that all counter values remain $< p$ through the iteration starting at c. ■

1.5 Periodic Orbits

The following result was proved in [2] using the checkpoint and the counter emulation methods:

Theorem 9 ([2]) *It is undecidable whether a given complete 2-DCM admits a periodic configuration.*

The next theorem states the analogous result for complete 3-RCM, as well as for 2-RCM, which are not necessarily complete.

Theorem 10 *It is undecidable whether a given complete 3-RCM admits a periodic configuration, and it is undecidable whether a given (not necessarily complete) 2-RCM admits a periodic configuration.*

PROOF. We first prove the result for complete 3-RCM. The construction in [12] shows that it is undecidable for a given 2-RCM $M = (S, 2, T)$ without periodic configurations and two given states s_1 and s_2 whether there are counter values n_1, n_2, m_1 and m_2 such that $(s_1, n_1, m_1) \vdash^* (s_2, n_2, m_2)$. By removing all transitions from state s_2 and all transitions into state s_1 we can assume without loss of generality that all configurations (s_1, n_1, m_1) and (s_2, n_2, m_2) are halting in M^{-1} and M, respectively. Using a similar idea as in the proof of Theorem 6 we effectively construct a 3-RCM $M' = (S \times \{+, -\}, 3, T')$ that simulates M forwards and backwards in time using states $(s, +)$ and $(s, -)$, respectively, and counters 1 and 2. The direction is switched at halting configurations. In addition, counter 3 is incremented at halting configurations, except when the state is s_1 or s_2.

Machine M' is clearly reversible and complete. Moreover, since M has no periodic configurations, the only periodic configurations of M' are those where M is simulated back and forth between states s_1 and s_2. Hence M' has a periodic orbit if and only if machine M can get from state s_1 into state s_2. This completes the proof for 3-RCM.

Using the counter emulation technique (Proposition 2) a 3-RCM can be converted into a 2-RCM and that conversion preserves periodic orbits. ■

The 2-RCM provided by the counter emulation technique is not complete. It seems likely that it can be modified to give a complete 2-RCM, but details remain to be worked out:

Conjecture 11 *It is undecidable whether a given complete* 2*-RCM admits a periodic configuration.*

2 Reversible Turing Machines

2.1 Definitions

The classical model of Turing machines consider machines with a moving head (a configuration is a triple $(s, z, c) \in S \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$). Following Kůrka [8], we consider machines with a moving tape as our base model to endow the space of configurations with a compact topology. Following [12], we define two kinds of instructions for a simpler syntactic characterization of local reversibility.

Let $\Delta = \{\leftarrow, \rightarrow\}$ be the set of directions with inverses $\leftarrow^{-1} = \rightarrow$ and $\rightarrow^{-1} = \leftarrow$. For all $\delta \in \Delta$ and $a \in \Sigma$, moving σ_δ and writing μ_a actions are defined for all

$c \in \Sigma^{\mathbb{Z}}$ and $z \in \mathbb{Z}$ as:

$$\sigma_\delta(c)(z) = \begin{cases} c(z+1) \text{ if } \delta = \rightarrow \\ c(z-1) \text{ if } \delta = \leftarrow \end{cases} \qquad \mu_a(c)(z) = \begin{cases} a & \text{if } z = 0 \\ c(z) \text{ if } z \neq 0 \end{cases}$$

A *Turing machine* M is a triple (S, Σ, T) where S is a finite set of states, Σ is a finite set of symbols, and $T \subseteq (S \times \Delta \times S) \cup (S \times \Sigma \times S \times \Sigma)$ is the transition table of the machine. A configuration $\mathfrak{c}$ of the machine is a pair (s, c) where $s \in S$ is a state and $c \in \Sigma^{\mathbb{Z}}$ is the content of the tape. The machine can transform a configuration $\mathfrak{c}$ in a configuration $\mathfrak{c}'$ in one step, noted as $\mathfrak{c} \vdash \mathfrak{c}'$, by applying an instruction $\iota \in T$. An instruction $(s, \delta, t) \in T \cap (S \times \Delta \times S)$ is a *move instruction* of the machine, it can be applied to any configuration (s, c), leading to the configuration $(t, \sigma_\delta(c))$. An instruction $(s, a, t, b) \in T \cap (S \times \Sigma \times S \times \Sigma)$ is a *matching instruction* of the machine, it can be applied to any configuration (s, c) where $c(0) = a$, leading to the configuration $(t, \mu_b(c))$. The transitive closure of $\vdash$ is noted as $\vdash^*$.

A configuration is *halting* if it cannot be transformed by the machine. A *complete* machine has no halting configuration.

A Turing machine (S, Σ, T) is a *deterministic Turing machine (DTM)* if at most one instruction can be applied from any configuration. Formally, the transition table must satisfy the following conditions:

$$\begin{aligned} (s, \delta, t) \in T \wedge (s', a', t', b') \in T &\Rightarrow s \neq s' \\ (s, \delta, t) \in T \wedge (s, \delta', t') \in T &\Rightarrow \delta = \delta' \wedge t = t' \\ (s, a, t, b) \in T \wedge (s, a, t', b') \in T &\Rightarrow t = t' \wedge b = b' \end{aligned}$$

The partial *global transition function* $G : S \times \Sigma^{\mathbb{Z}} \to S \times \Sigma^{\mathbb{Z}}$ of a DTM maps a configuration to the unique transformed configuration, that is for all $\mathfrak{c} \in S \times \Sigma^{\mathbb{Z}}$, either $\mathfrak{c}$ is halting and $G(\mathfrak{c})$ is undefined, or $\mathfrak{c}$ is non-halting and $G(\mathfrak{c}) = \mathfrak{c}'$ where $\mathfrak{c}'$ is the unique configuration such that $\mathfrak{c} \vdash \mathfrak{c}'$.

The space $S \times \Sigma^{\mathbb{Z}}$ of configurations is endowed with a compact and metrizable topology, obtained as the product of the discrete topology on S by the infinite product of the discrete topology on Σ. See, for example, [8] for more details. The global transition function of a DTM is a continuous (partial) function, leading to uniformity properties:

Lemma 12 *If all configurations of a DTM are periodic or mortal then there is a uniform bound n such that for all configurations (s, c) either $G^n(s, c)$ is undefined or $G^t(s, c) = (s, c)$ for some $0 < t < n$. In particular, a periodic DTM is uniformly periodic and a mortal DTM is uniformly mortal.*

PROOF. For every $n > 0$ let $U_n = \{(s, c) \mid G^n(s, c) = (s, c) \text{ or } G^n(s, c) \text{ undef}\}$ be the set of configurations that are mortal or periodic at time n. Sets U_n are open so $U_1, U_2, \ldots$ is an open cover of the compact set of all configurations. It has a finite subcover. ■

One might think that periodicity characterizes a different set of machines if one considers Turing machines with a moving head instead of a moving tape but it is not the case. The global transition function with moving head $H : S \times \mathbb{Z} \times \Sigma^{\mathbb{Z}} \to S \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$ is defined so that for each $(s, z, c) \in S \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$, $H(s, z, c) = (s', z', c')$ where $G(s, \sigma^z_{\rightarrow}(c)) = (s', \sigma^{z'}_{\rightarrow}(c'))$. A DTM is periodic with moving head if for each configuration $\mathfrak{c}$, there exists $t \in \mathbb{N}$ such that $H^t(\mathfrak{c}) = \mathfrak{c}$.

Lemma 13 *A DTM (S, Σ, T) with $|\Sigma| \geq 2$ is (uniformly) periodic if and only if it is (uniformly) periodic with moving head.*

PROOF. Assume that Σ has at least two elements. For each $t \in \mathbb{N}$ and $(s, z, c) \in S \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$, $H^t(s, z, c) = (s', z', c')$ where $G^t(s, \sigma^z_{\rightarrow}(c)) = (s', \sigma^{z'}_{\rightarrow}(c'))$. Thus, if $H^t(s, 0, c) = (s, 0, c)$ then $G^t(s, c) = (s, c)$, so periodicity with the moving head implies periodicity. By Lemma 12 this implies uniform periodicity.

Conversely, let $G^t = \mathrm{Id}$. By definition, $H^t(s, z, c) = (s, z', c')$ for some z' such that $\sigma^z_{\rightarrow}(c) = \sigma^{z'}_{\rightarrow}(c')$. Moreover, as the machine acts locally, for all d and k such that $c_{|[z-t,z+t]} = d_{|[k-t,k+t]}$, $H^t(s, k, d) = (s, k+z'-z, d')$ where $d' = \sigma^{z'-z}_{\rightarrow}(d')$. If $z'-z \neq 0$, one might choose d such that $d(k+t(z'-z)) \neq d(k+(t+1)(z'-z))$, contradicting the hypothesis. Thus, $H^t = \mathrm{Id}$. ■

So there is no difference between the moving head and the moving tape modes as far as periodicity and immortality properties are considered. For this reason we assume the moving tape mode throughout the rest of the paper.

The reverse of an instruction is defined as follows: $(s, \delta, t)^{-1} = (t, \delta^{-1}, s)$ and $(s, a, t, b)^{-1} = (t, b, s, a)$. The reverse T^{-1} of a transition table T is defined as $T^{-1} = \{\iota^{-1} | \iota \in T\}$. The *reverse* of Turing machine $M = (S, \Sigma, T)$ is the machine $M^{-1} = (S, \Sigma, T^{-1})$. Observe that $\mathfrak{c} \vdash \mathfrak{c}'$ in M if and only if $\mathfrak{c}' \vdash \mathfrak{c}$ in M^{-1}. A *reversible Turing machine (RTM)* is a deterministic Turing machine whose reverse is deterministic. Clearly a DTM is reversible if and only if its global transition function is one-to-one.

Example 14 *The complete DTM $(\{l, l', r, r'\}, \{a, b\}, T)$ with the following T is reversible: $\{(l, b, l', b), (l, a, r', a), (r, a, r', b), (r, b, l', a), (l', \leftarrow, l), (r', \rightarrow, r)\}$. Its reverse is the complete DTM $(\{l, l', r, r'\}, \{a, b\}, T')$ with the following T':*

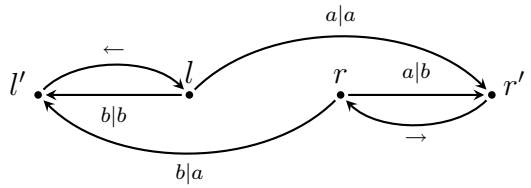


Fig. 2. a complete RTM

$\{(l, \rightarrow, l'), (r, \leftarrow, r'), (r', a, l, a), (l', b, l, b), (l', a, r, b), (r', b, r, a)\}$. *The first machine is depicted on Fig. 2.* ◇

Lemma 15 *It is decidable whether a given Turing machine is reversible.*

PROOF. It is sufficient to syntactically check the transition table. ■

Lemma 16 *The reverse of a mortal RTM is mortal.*

PROOF. The uniform bound is valid for both the mortal RTM and its reverse. ■

In contrast to counter machines we have the following.

Lemma 17 *The reverse of a complete RTM is a complete RTM. In particular, a complete RTM is surjective.*

PROOF. A DTM is complete if and only if $n|\Sigma|+m = |S||\Sigma|$ where n and m are the numbers of move and matching instructions, respectively. The claim follows from the fact that M and M^{-1} always have the same numbers of move and matching instructions. ■

2.2 Programming RTM

In order to precisely define and test properties of our construction for theorem 20, we designed a specific programming language called Gni. In order to program in Gni, we developed the Gnirut toolkit [13]: a software package consisting of a compiler, an interpreter and various tools to program (reversible) Turing machines and test them. A precise description of the language is part of the toolkit. Let us here provide a simplified description of an extended language near enough from Gni that the reader can itself write the corresponding programs.

2.2.1 Statements

A Gni program is a sequence of statements. Each statement is written on a separate line. In a Gni program, states and symbols are represented by identifiers. The set of states and symbols of the machine defined by a program is the set of states and symbols appearing in its statements. The syntax of basic statements is the following:

s. ←, *s'* denotes the move instruction $(s, \leftarrow, s')$;
s. →, *s'* denotes the move instruction $(s, \rightarrow, s')$;
s. a ⊢ b, *s'* denotes the matching instruction (s, a, s', b).

To simplify the writing of matching statements, several matching with the same initial state can be merged into a compound statement. With the same philosophy, default behavior can be defined. The syntax of compound statements is the following:

s. a ⊢ b, t | b ⊢ c, u | d ⊢ e, v denotes the sequence of matching instructions (s, a, b, t), (s, b, c, u), (s, d, e, v);
s. a ⊢ b, *s'* else t same as before but with a default behavior: on any letter not defined in the compound matching instructions, do not modify the letter but jump to state t.

Finally, the most extended type of statement combines both matching and move statements to branch into different states depending on a local word around the head. An extended statement looks like a matching statement in which each pair of letters of a matching case (like in a ⊢ b, t) is replaced by a pair of words of the same size, one letter on each being underlined (like in $\underline{\texttt{a}}\texttt{ba} \vdash \texttt{bb}\underline{\texttt{b}}, t$). The meaning of such a matching case is the following: starting with the head on the underlined position in the first word, check if the tape contains the first word and if it is the case, then replace it by the second word, place the head in the underlined position of the second word and enter the given state.

2.2.2 Macros

As such, the language is clearly sufficient to encode any machine but it can be boring to repeat over and over the same patterns, copies of the same submachine performing a given task like *go to the first x on the right*. To help on this, the language provides a macro definition system.

Defining a macro is really the same as defining the main machine. The same instructions are used. In both cases a Turing machine is defined. But for macro definitions all states loose their names after definition but the ones identified as

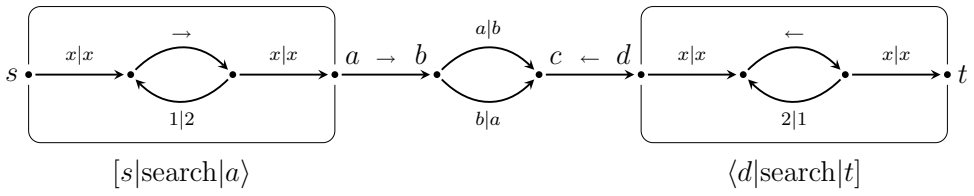


Fig. 3. a sample compiled RTM with 10 states

relevant to the outside world. Moreover, the machine associated to the macro is not added to the current main machine but stored for future use. Once the macro defined, to use it, add a fresh copy of the machine states in the current environment.

Indentation plays a big role in macro definition. After the `def` starting the macro definition, all the lines corresponding to that macro should be indented by the same amount of space. When the indentation goes back to the original level, the macro definition ends.

The syntax for macro definition and usage is the following:

def $[i_1, \ldots, i_m|\text{toto}|o_1, \ldots, o_n\rangle$: begins the definition of a macro called toto; the only states that will be visible from outside the macro are $i_1, \ldots, i_m$ and $o_1, \ldots, o_n$. There is no technical distinction between i and o but it is good practice to consider i as input states and o as output states, to facilitate reading of source code;

$[a, b|\text{titi}|c, d, e\rangle$ inserts a copy of the macro machine titi in the current machine definition: the fresh copy of the machine titi will use a, b as input states for its i_1, i_2 and c, d, e as output states for its o_1, o_2, o_3;

$\langle o_1, \ldots, o_n|\text{toto}|i_1, \ldots, i_m]$ inserts a copy of the reverse of the machine toto, provided that toto is reversible.

Example 18 *The following code defines a macro and uses it and its reverse:*

```
1  def [s|search|t⟩ :
2      s. x ⊢ x, u
3      u. →, r
4      r. 1 ⊢ 2, u | x ⊢ x, t
5
6   [s|search|a⟩
7   a. →, b
8   b. a ⊢ b, c | b ⊢ a, c
9   c. ←, d
10  ⟨d|search|t]
```

The machine defined by this program is depicted on Fig. 3 where macros usage is highlighted. ◇

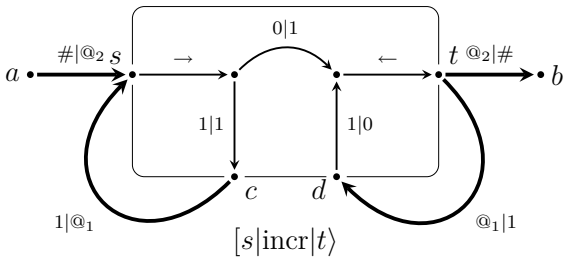


Fig. 4. a sample compiled recursive RTM with 8 states

2.2.3 Recursion

The last construction of the Gni language is the key part of Hooper's [4] construction: recursion via function calls in the style of stack based machines using the tape as a place to push entry point information. The principle is simple: to call a given submachine, first write down on the tape a letter identifying the current state, then enter the input state of the submachine; when the submachine returns, read the entry point on the tape to know in which state to return. Notice that, for the construction to work, the head should be at the same position on the tape when entering and when leaving the submachine. Whereas the Gni language provides syntactic sugar to use recursion, the programmer has to ensure itself that the position constraint is fulfilled. The syntax is the following:

fun $[i_1, \ldots, i_m|\text{toto}|o_1, \ldots, o_n\rangle$: defines a callable function: the syntax and principle are the same as for defining macros but the obtained machine is used through calls and just one copy of it will be put into the main machine;

call $[i_1, \ldots, i_m|\text{toto}|o_1, \ldots, o_n\rangle$ from a inserts a function call: when entering states i_1 to i_m, the machine will push entry point information on the tape, replacing the letter a that is written on it; on return from the call it will pop the entry point information, replace it by a and change state to o_1,...,o_n depending on the return state.

Example 19 *The following code recursively increment a binary integer:*

```
1  fun [s|incr|t⟩ :                    8  call [a|incr|b⟩ from #
2      s. →, r
3      r. 0 ⊢ 1, b | 1 ⊢ 1, c
4      call [c|incr|d⟩ from 1
5      d. 1 ⊢ 0, b
6      b. ←, t
7
```

The machine defined by this program is depicted on Fig. 4 where transitions generated for recursion are highlighted. ◇

2.3 Undecidability of the Immortality Problem

2.3.1 Principle

One might consider using the same approach as for Theorem 3: combine a proof of the undecidability of the immortality problem for Turing machines with a simulation of DTM by RTM. Unfortunately this cannot work directly as the classical constructions [10,1] do bookkeeping by emulating a stack on the tape. By searching the extremity of the stack to push, the machine performs an unbounded search, introducing infinite orbits: the simulation does not preserve mortality. Thus, we have to consider a new proof along the lines of Hooper's [4] proof but for the restricted class of RTM.

For a given 2-RCM without periodic configurations, and given initial state s_0, we effectively construct a reversible Turing machine that is mortal if and only if the 2-RCM halts from the initial configuration $(s_0, 0, 0)$. The Theorem then follows from [12], where it was shown that the halting problem is undecidable for 2-RCM. Note that our additional constraint that the 2-RCM has no periodic configurations can be easily established combining bounded past and counter emulation techniques that do preserve mortality.

As a first step we do a fairly standard simulation of a 2-RCM by a RTM. Configuration (s, a, b) where s is a state and $a, b \in \mathbb{N}$ is represented as a block "`@1`^{a}`x2`^{b}`y`" of length $a + b + 3$, and the Turing machine is positioned on the symbol "`@`" in state s. A simulation of one move of the RCM consists of (1) finding delimiters "`x`" and "`y`" on the right to check if either of the two counters is zero, and (2) incrementing or decrementing the counters as determined by the RCM. The RTM is then returned to the beginning of the block in the new state of the RCM. If the RCM halts then also the RTM halts. All this can be done reversibly.

The RTM constructed as outline above has the problem that it has immortal configurations even if the RCM halts. These are due to the unbounded searches for delimiter symbols "`@`", "`x`" or "`y`". Searches are needed when testing whether the second counter is zero, as well as whenever either counter is incremented or decremented.

Unbounded searches lead to infinite searches if the symbol is not present in the configuration. (For example, searching to the right for symbol "`x`" when the tape contains "`@111`...".) To prevent such infinite searches we follow the idea of [4], also employed in [2]. Instead of a straightforward search using a loop, the search is done by performing a recursive call to the counter machine from its initial configuration $(s_0, 0, 0)$. More precisely, we first make a bounded search of length three to see if the delimiter is found within next three symbols. If the delimiter is not found, we start a recursive simulation of the RCM by writing

"`@xy`" over the next three symbols, step on the new delimiter symbol "`@`", and enter the initial state s_0. This begins a nested simulation of the RCM.

In order to be able to continue the higher level execution after returning from the recursive search, the present state of the TM needs to be written on the tape when starting the recursive call. For this purpose we increase the tape alphabet by introducing several variants "$\texttt{@}_\alpha$" of the start delimiter "`@`". Here α is the Turing machine state at the time the search was begun. When returning from a successful recursive search, the higher level computation can pick up from where it left off by reading the state α from the delimiter "$\texttt{@}_\alpha$".

If the recursive search procedure finds the delimiter this is signaled by reversing the search. Once returned to the beginning, the three symbol initial segment "`@xy`" is moved three positions to the right and the process is repeated. The repeated applications of recursive searches, always starting the next search three positions further right, will eventually bring the machine on the delimiter it was looking for, and the search is completed.

On the other hand, if the RCM halts during a recursive search then the RTM halts. This always happens when a sufficiently long search is performed using a RCM that halts from its initial configuration.

2.3.2 Construction

We explain here the details of the construction and how it should be precisely done so that the obtained RTM verifies all hypothesis. We provide the reader with two main arguments. First, we describe the syntactic form of the part of the tape already visited by the head of the machine since the beginning of the computation. Then, we incrementally describe precisely the machine by giving building blocks and their invariants.

2.3.2.1 Terra cognita The machine is constructed in such a way that the *terra cognita*, the part of the configuration that has been visited by the machine head since the beginning of the computation, always (re)enters a k-depth well-parenthesis word S as described by the following grammar in a bounded number of steps.

$$
\begin{aligned}
S &\leftarrow S_1 \mid S_2 \mid S_{\underline{1}2} \mid S_{1\underline{2}} \\
S_1 &\leftarrow \mathtt{1}^* P_k \mathtt{1}^* \mid \mathtt{@}_\alpha \mathtt{1}^* P_k \mathtt{1}^* \\
S_2 &\leftarrow \mathtt{2}^* P_k \mathtt{2}^* \mid \mathtt{2}^* P_k \mathtt{2}^* \mathtt{y} \\
S_{\underline{1}2} &\leftarrow \mathtt{1}^* P_k \mathtt{1}^* \mathtt{x} \mathtt{2}^* \mid \mathtt{@}_\alpha \mathtt{1}^* P_k \mathtt{1}^* \mathtt{x} \mathtt{2}^* \mid \mathtt{1}^* P_k \mathtt{1}^* \mathtt{x} \mathtt{2}^* \mathtt{y} \\
S_{1\underline{2}} &\leftarrow \mathtt{1}^* \mathtt{x} \mathtt{2}^* P_k \mathtt{2}^* \mid \mathtt{@}_\alpha \mathtt{1}^* \mathtt{x} \mathtt{2}^* P_k \mathtt{2}^* \mid \mathtt{1}^* \mathtt{x} \mathtt{2}^* P_k \mathtt{2}^* \mathtt{y} \\
P_k &\leftarrow \mathtt{@}_\alpha \mathtt{1}^* P_{k-1} \mathtt{1}^* \mathtt{x} \mathtt{2}^* \mathtt{y} \mid \mathtt{@}_\alpha \mathtt{1}^* \mathtt{x} \mathtt{2}^* P_{k-1} \mathtt{2}^* \mathtt{y} \\
P_0 &\leftarrow \mathtt{@}_\alpha \underline{\mathtt{x}} \mathtt{y}
\end{aligned}
$$

Moreover, the only local transformations the head does on *terra cognita* are:

- replace 111 or 222 by $\mathtt{@}_\alpha\mathtt{xy}$ *(add a new level)*;
- replace 1x2 or 1xy by 11x *(increment counter 1)*;
- replace 2y1 or 2y2 by 22y *(increment counter)*;
- replace $\mathtt{@}_\alpha\mathtt{xy}$ by 111 or 222 as expected by surroundings *(remove a level)*;
- replace 11x by 1x2 or 1xy *(decrement counter 1)*;
- replace 22y by 2y1 or 2y2 as expected by surroundings *(decrement counter)*;

These transformations ensure *terra cognita* is always well formed. If the machine encounters a violation of the grammar when discovering new parts of the tape, it halts.

2.3.2.2 Constructing the machine We start with some hypothesis on checking and from there build counter machine simulator and effective checking machine. All the constructed machines are reversible and halt on unspecified configurations.

Assume that when it returns, $[s|\text{check}_1|t\rangle$ verifies $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^m\mathtt{x} \vdash \underline{\mathtt{@}_\alpha}\mathtt{1}^m\mathtt{x}, t$. If it does not return, either it diverges on $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^\omega$ or it halts.

Symmetrically, assume that when it returns, $[s|\text{check}_2|t\rangle$ verifies $s.\ \underline{\mathtt{x}}\mathtt{2}^n\mathtt{y} \vdash \underline{\mathtt{x}}\mathtt{2}^n\mathtt{y}, t$. If it does not return, either it diverges on $s.\ \underline{\mathtt{x}}\mathtt{2}^\omega$ or it halts.

2.3.2.3 Bounded search Bounded search is the key part of the construction calling checking recursively, described on source code 1.

When it does not halt, $[s|\text{search}_1|t_0, t_1, t_2\rangle$ verifies $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^m\mathtt{x} \vdash \mathtt{@}_\alpha\mathtt{1}^m\underline{\mathtt{x}}, t_k$ where $k = m \mod 3$ or diverges into check_1 on $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^\omega$. Notice that if the first call to

check$_1$ in the loop returns, all subsequent calls will also return and the search will eventually return.

Symmetrically, when it does not halt, $[s|\text{search}_2|t_0, t_1, t_2\rangle$ verifies $s.\ \underline{\mathtt{x}}\mathtt{2}^n\mathtt{y} \vdash \mathtt{x}\mathtt{2}^n\underline{\mathtt{y}}, t_k$ where $k = n \mod 3$ or diverges into check$_2$ on $s.\ \underline{\mathtt{x}}\mathtt{2}^\omega$. Notice that if the first call to check$_2$ in the loop returns, all subsequent calls will also return and the search will eventually return.

2.3.2.4 RCM simulator The counter machine simulation is quite standard, one has to ensure that bounded searches are used and that the instruction machines are reversible. Testing instructions are described on source code 2, incrementation/decrementation of the second counter on source code 3 and incrementation/decrementation of the first counter on source code 4.

Test counter 1 $[s|\text{test1}|z, p\rangle$ verifies $s.\ \underline{\mathtt{@}_\alpha}\mathtt{x} \vdash \underline{\mathtt{@}_\alpha}\mathtt{x}, z$ and $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1} \vdash \underline{\mathtt{@}_\alpha}\mathtt{1}, p$.

Test counter 2 $[s|\text{test2}|z, p\rangle$ verifies $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^n\mathtt{xy} \vdash \underline{\mathtt{@}_\alpha}\mathtt{1}^n\mathtt{xy}, z$ and $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^n\mathtt{x2} \vdash \underline{\mathtt{@}_\alpha}\mathtt{1}^n\mathtt{x2}, p$ and diverges into check$_1$ on $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^\omega$.

Increment/decrement counter The principle is the same as for testing: we use bounded search and ensure reversibility. Notice that, to ensure reversibility, each instruction requires two variants depending on the surroundings.

RCM simulator After initializing the tape, just use the instruction machines using one small trick on test instructions: each state reached via a test instruction should be coupled to a reverse test instruction to ensure reversibility (such a coupling is always possible if the simulated machine is reversible). Every possible collision is an output of the simulator. Source code 5 provide a complete example of a sample RCM simulation.

2.3.2.5 Checking Checking simply consists in simulating CM until collision and then reverting back to initial position as described on source code 6.

When it does not halt, $[s|\text{check}_1|t\rangle$ verifies $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^m\mathtt{x} \vdash \underline{\mathtt{@}_\alpha}\mathtt{1}^m\mathtt{x}, t$ or diverges on (or halts on a prefix of) $s.\ \underline{\mathtt{@}_\alpha}\mathtt{1}^\omega$. If it diverges, it executes $[s|\text{RCM}_\alpha|co_1, co_2, \ldots\rangle$ on segments of unbounded size, thus the RCM does not halt.

Symmetrically, when it does not halt, $[s|\text{check}_2|t\rangle$ verifies $s.\ \underline{\mathtt{x}}\mathtt{2}^n\mathtt{y} \vdash \underline{\mathtt{x}}\mathtt{2}^n\mathtt{y}, t$ or diverges on (or halts on a prefix of) $s.\ \underline{\mathtt{x}}\mathtt{2}^\omega$. If it diverges, it executes $[s|\text{RCM}_\alpha|co_1, co_2, \ldots\rangle$ on segments of unbounded size, thus the RCM does not halt.

```
def [s|search1|t0, t1, t2⟩ :
    s. @α ⊢ @α, l
    l. →, u
    u. x ⊢ x, t0
     | 1x ⊢ 1x, t1
     | 11x ⊢ 11x, t2
     | 111 ⊢ 111, c
    call [c|check1|p⟩ from 1
    p. 111 ⊢ 111, l
```

```
def [s|search2|t0, t1, t2⟩ :
    s. x ⊢ x, l
    l. →, u
    u. y ⊢ y, t0
     | 2y ⊢ 2y, t1
     | 22y ⊢ 22y, t2
     | 222 ⊢ 222, c
    call [c|check2|p⟩ from 2
    p. 222 ⊢ 222, l
```

Source code 1. Bounded search

```
def [s|test1|z, p⟩ :
    s. @αx ⊢ @αx, z
     | @α1 ⊢ @α1, p

def [s|endtest2|z, p⟩ :
    s. xy ⊢ xy, z
     | x2 ⊢ x2, p
```

```
def [s|test2|z, p⟩ :
    [s|search1|t0, t1, t2⟩
    [t0|endtest2|z0, p0⟩
    [t1|endtest2|z1, p1⟩
    [t2|endtest2|z2, p2⟩
    ⟨z0, z1, z2|search1|z]
    ⟨p0, p1, p2|search1|p]
```

Source code 2. Testing instructions

```
def [s|mark1|t, co⟩ :
    s. y1 ⊢ 2y, t
     | yx ⊢ yx, co

def [s|endinc1|t, co⟩ :
    [s|search2|r0, r1, r2⟩
    [r0|mark1|t0, co0⟩
    [r1|mark1|t1, co1⟩
    [r2|mark1|t2, co2⟩
    ⟨t2, t0, t1|search2|t]
    ⟨co0, co1, co2|search2|co]

def [s|inc21|t, co⟩ :
    [s|search1|r0, r1, r2⟩
    [r0|endinc1|t0, co0⟩
    [r1|endinc1|t1, co1⟩
    [r2|endinc1|t2, co2⟩
    ⟨t0, t1, t2|search1|t]
    ⟨co0, co1, co2|search1|co]

def [s|dec21|t⟩ :
    ⟨s, co|inc21|t]
```

```
def [s|mark2|t, co⟩ :
    s. y2 ⊢ 2y, t
     | yx ⊢ yx, co

def [s|endinc2|t, co⟩ :
    [s|search2|r0, r1, r2⟩
    [r0|mark2|t0, co0⟩
    [r1|mark2|t1, co1⟩
    [r2|mark2|t2, co2⟩
    ⟨t2, t0, t1|search2|t]
    ⟨co0, co1, co2|search2|co]

def [s|inc22|t, co⟩ :
    [s|search1|r0, r1, r2⟩
    [r0|endinc2|t0, co0⟩
    [r1|endinc2|t1, co1⟩
    [r2|endinc2|t2, co2⟩
    ⟨t0, t1, t2|search1|t]
    ⟨co0, co1, co2|search1|co]

def [s|dec22|t⟩ :
    ⟨s, co|inc22|t]
```

Source code 3. Increment/decrement counter 2

```
def [s|pushinc1|t, co⟩ :
  s. x2 ⊢ 1x, c
  | xy1 ⊢ 1xy, pt
  | xyx ⊢ 1yx, pco
  [c|endinc1|pt0, pco0⟩
  pt0. →, t0
  t0. 2 ⊢ 2, pt
  pt. ←, t
  pco0. x ⊢ 2, pco
  pco. ←, zco
  zco. 1 ⊢ x, co
```

```
def [s|pushinc2|t, co⟩ :
  s. x2 ⊢ 1x, c
  | xy2 ⊢ 1xy, pt
  | xyy ⊢ 1yy, pco
  [c|endinc2|pt0, pco0⟩
  pt0. →, t0
  t0. 2 ⊢ 2, pt
  pt. ←, t
  pco0. x ⊢ 2, pco
  pco. ←, zco
  zco. 1 ⊢ x, co
```

```
def [s|inc11|t, co⟩ :
  [s|search1|r0, r1, r2⟩
  [r0|pushinc1|t0, co0⟩
  [r1|pushinc1|t1, co1⟩
  [r2|pushinc1|t2, co2⟩
  ⟨t2, t0, t1|search1|t]
  ⟨co0, co1, co2|search1|co]
```

```
def [s|inc12|t, co⟩ :
  [s|search1|r0, r1, r2⟩
  [r0|pushinc2|t0, co0⟩
  [r1|pushinc2|t1, co1⟩
  [r2|pushinc2|t2, co2⟩
  ⟨t2, t0, t1|search1|t]
  ⟨co0, co1, co2|search1|co]
```

```
def [s|dec11|t⟩ :
  ⟨s, co|inc11|t]
```

```
def [s|dec12|t⟩ :
  ⟨s, co|inc12|t]
```

Source code 4. Increment/decrement counter 1

```
def [s|init1|r⟩ :
  s. →, u
  u. 11 ⊢ xy, e
  e. ←, r
```

```
def [s|init2|r⟩ :
  s. →, u
  u. 22 ⊢ xy, e
  e. ←, r
```

```
def [s|RCM1|co1, co2⟩ :
  [s|init1|s0⟩
  [s0|test1|s1z, n⟩
  [s1|inc11|s2, co1⟩
  [s2|inc21|s3, co2⟩
  [s3|test1|n', s1p⟩
  ⟨s1z, s1p|test1|s1]
```

```
def [s|RCM2|co1, co2⟩ :
  [s|init2|s0⟩
  [s0|test1|s1z, n⟩
  [s1|inc12|s2, co1⟩
  [s2|inc22|s3, co2⟩
  [s3|test1|n', s1p⟩
  ⟨s1z, s1p|test1|s1]
```

Source code 5. RCM simulator

```
fun [s|check1|t⟩ :
  [s|RCM1|co1, co2, ...⟩
  ⟨co1, co2, ...|RCM1|t]
```

```
fun [s|check2|t⟩ :
  [s|RCM2|co1, co2, ...⟩
  ⟨co1, co2, ...|RCM2|t]
```

Source code 6. Checking with RCM simulator

2.3.3 Proof

Theorem 20 *It is undecidable whether a given RTM is immortal.*

PROOF. For a given 2-RCM without periodic configurations, and given initial state s_0, we consider its RTM simulator.

If the initial configuration $(s_0, 0, 0)$ is immortal in the RCM then the RTM has a non-halting simulation of the RCM. So the RTM is not mortal.

Conversely, suppose that the RCM halts in k steps but the RTM has an immortal configuration. The only way for the RTM not to halt is to properly simulate the RCM from some configuration (s, a, b), where the possibilities $a = \infty$ and $b = \infty$ have to be taken into account. Since the RCM has no periodic configurations, one of the two counters necessarily obtains arbitrarily large values during the computation. But this leads to arbitrarily long recursive searches, which is not possible since each such search halts within a bounded number of steps. ■

Remark 21 *The RTM constructed in the proof has no periodic configurations. So the undecidability of the immortality problem holds among RTM without any periodic configurations.*

Remark 22 *Add to the 2-RCM a new looping state s_1 in which the first counter is incremented indefinitely. We can also assume without loss of generality that the 2-RCM halts only in state s_2. Then the RTM constructed in the proof has computation $(s_1, c_1) \vdash^* (s_2, c_2)$ for some $c_1, c_2 \in \Sigma^{\mathbb{Z}}$ if and only if the 2-RCM halts from the initial configuration $(s_0, 0, 0)$.*

These detailed observations about the proof will be used later in the proofs of Theorems 23 and 24.

2.4 Undecidability of the Periodicity Problem

Theorem 23 *It is undecidable whether a given complete RTM is periodic.*

PROOF. For a given RTM $A = (S, \Sigma, T)$ we effectively construct a complete RTM $A' = (S \times \{+, -\}, \Sigma, T')$ that is periodic if and only if every configuration of A is either periodic or mortal. States $(s, +)$ and $(s, -)$ are used to represent A in state s running forwards or backwards in time, respectively. In a halting configuration the direction is switched. More precisely, let f and f^{-1} be the

local transition functions of A and A^{-1}, respectively. Then the transition table T' is constructed so that the local transition function f' of A' is

$$f'((s,+),a) = \begin{cases} ((t,+),\delta) & \text{if } f(s,a) = (t,\delta) \\ ((t,+),b) & \text{if } f(s,a) = (t,b) \\ ((s,-),a) & \text{if } f(s,a) = \perp \end{cases}$$

$$f'((s,-),a) = \begin{cases} ((t,-),\delta) & \text{if } f^{-1}(s,a) = (t,\delta) \\ ((t,-),b) & \text{if } f^{-1}(s,a) = (t,b) \\ ((s,+),a) & \text{if } f^{-1}(s,a) = \perp \end{cases}$$

It is easy to see that A' is complete and reversible. If all configurations are periodic or mortal in A then they are also periodic or mortal in A^{-1}. It follows that all configurations are periodic in A'. Conversely, suppose that (s,c) is a configuration of A that is neither periodic nor mortal. Then $((s,+),c)$ is not periodic in A'. We conclude that A' is periodic if and only if all configurations of A are either periodic or mortal. According to Remark 21, we may assume that A has no periodic configurations. In this case periodicity of A' is equivalent to mortality of A, and the result follows from Theorem 20. ■

2.5 Periodic Orbits

Theorem 24 *It is undecidable whether a given (non-complete) RTM admits a periodic configuration.*

PROOF. Remark 22 pointed out that it is undecidable for a given RTM $A = (S, \Sigma, T)$ without periodic configurations, and two given states $s_1, s_2 \in S$ whether there are configurations (s_1, c_1) and (s_2, c_2) such that $(s_1, c_1) \vdash^* (s_2, c_2)$. By removing all transitions from state s_2 and all transitions into state s_1 we can assume without loss of generality that all configurations (s_1, c_1) and (s_2, c_2) are halting in A^{-1} and A, respectively. Using a similar idea as in the proof of Theorem 23 we effectively construct an RTM $A' = (S \times \{+,-\}, \Sigma, T')$ in which A is simulated forwards and backwards in time using states $(s,+)$ and $(s,-)$, respectively. But now the direction is swapped from "-" to "+" only in state s_1, and from "+" to "-" in state s_2. In other halting situations of A, also A' halts. Clearly $((s_1,+),c_1)$ is periodic in A' if and only if $(s_1,c_1) \vdash^* (s_2,c_2)$ for some $c_2 \in \Sigma^{\mathbb{Z}}$. No other periodic orbits exist in A'. ■

Theorem 25 *It is undecidable whether a given complete DTM admits a periodic configuration.*

PROOF. In [2] a complete DTM over the binary tape alphabet was provided that does not have any periodic configurations. This easily gives an analogous DTM for any bigger tape alphabet. For a given RTM $A = (S, \Sigma, T)$ we effectively construct a complete DTM that has a periodic configuration if and only if A has a periodic configuration. The result then follows from Theorem 24. Let $B = (S', \Sigma, T')$ be the fixed complete DTM without periodic configurations from [2], $S \cap S' = \emptyset$. The complete DTM we construct has state set $S \cup S'$ and its transitions includes $T \cup T'$, and in addition a transition into a state $s' \in S'$ whenever A halts. It is clear that the only periodic configurations are those that are periodic already in A. ■

Conjecture 26 *A complete RTM without a periodic point exists. Moreover, it is undecidable whether a given complete RTM admits a periodic configuration.*

3 Reversible Cellular Automata

3.1 Definitions

A *one-dimensional cellular automaton* A is a triple (S, r, f) where S is a finite *state set*, $r \in \mathbb{N}$ is the *neighborhood radius* and $f : S^{2r+1} \longrightarrow S$ is the *local update rule* of A. Elements of $\mathbb{Z}$ are called cells, and a configuration of A is an element of $S^{\mathbb{Z}}$ that assigns a state to each cell. Configuration c is turned into configuration c' in one time step by a simultaneous application of the local update rule f in the radius r neighborhood of each cell:

$$c'(i) = f(c(i-r), c(i-r+1), \ldots, c(i+r-1), c(i+r)) \text{ for all } i \in \mathbb{Z}.$$

Transformation $G : c \mapsto c'$ is the *global transition function* of A. The Curtis-Hedlund-Lyndom-theorem states that a function $S^{\mathbb{Z}} \longrightarrow S^{\mathbb{Z}}$ is a global transition function of some CA if and only if it is continuous and commutes with the shift σ, defined by $\sigma(c)_i = c_{i+1}$ for all $c \in S^{\mathbb{Z}}$ and $i \in \mathbb{Z}$.

Cellular automaton A is called *reversible* if the global function G is bijective and its inverse G^{-1} is a CA function. We call A *injective*, *surjective* and *bijective* if G is injective, surjective and bijective, respectively. Injectivity implies surjectivity, and bijectivity implies reversibility. See [5] for more details on these classical results.

3.2 Undecidability of the Immortality Problem

Let some states of a CA be identified as halting. Let us call a configuration c *halting* if $c(i)$ is a halting state for some i. We call c *locally halting* if $c(0)$ is a halting state. These two definitions reflect two different ways that one may use to define an accepting computation in CA: either acceptance happens when a halting state appears somewhere, in an unspecified cell, or one waits until a halting state shows up in a fixed, predetermined cell. A configuration c is immortal (locally immortal) for G if $G^n(c)$ is not halting (locally halting, respectively) for any $n \geq 0$. CA function G is immortal (locally immortal) if there exists an immortal (locally immortal) configuration.

Theorem 27 ([6]) *It is undecidable whether a given reversible one-dimensional CA is immortal (locally immortal).*

3.3 Undecidability of the Periodicity Problem

In cellular automata periodicity and uniform periodicity are equivalent. Indeed, suppose that a period n that is common to all configurations does not exist. Then for every $n \geq 1$ there is $c_n \in S^{\mathbb{Z}}$ such that $G^n(c_n) \neq c_n$. Each c_n has a finite segment p_n of length $2rn + 1$ that is mapped in n steps into a state that is different from the state in the center of p_n. Configuration c that contains a copy of p_n for all n, satisfies $G^n(c) \neq c$ for all n, and hence such c is not periodic.

Theorem 28 *It is undecidable whether a given one-dimensional CA is periodic.*

PROOF. For a given complete reversible Turing machine $M = (S, \Sigma, T)$ we effectively construct a one-dimensional reversible CA $A = (Q, 2, f)$ that is periodic if and only if M is periodic. The state set

$$Q = \Sigma \times ((S \times \{+, -\}) \cup \{\leftarrow, \rightarrow\})$$

consists of two tracks: The first track stores elements of the tape alphabet Σ and it is used to simulate the content of the tape of the Turing machine, while the second track stores the current state of the simulated machine at its present location, and arrows $\leftarrow$ and $\rightarrow$ in other positions pointing towards the position of the Turing machine on the tape. The arrows are needed to prevent several Turing machine heads accessing the same tape location and interfering with each other's computation. The state is associated a symbol

'+' or '-' indicating whether the reversible Turing machine is being simulated forwards or backwards in time.

The local update rule f only can change the state of a cell whose radius-one neighborhood contains a Turing machine state on the second track. Let i be a cell that contains Turing machine state on the second track, and let $i + \delta$ be the position of the Turing machine after its next move, where $\delta \in \{-1, 0, 1\}$. Note that the next move is forward or backward in time depending on whether the symbol associated with the state is '+' or '-', respectively.

The move consists of possibly changing the tape symbol on the first track in position i, writing the new state of M in cell $i + \delta$ and writing a left or right arrow in position i depending on whether $\delta = -1$ or $\delta = +1$, respectively. These instructions are as indicated by the local transition function of the Turing machine.

But the move is executed in the CA only if the tape looks locally correct, that is, the nearby cells have arrows pointing towards the TM. More precisely, the second track of cells $i - 1$ and $i + 1$ must contain a right and left arrow, respectively, and

- if $\delta = -1$ then position $i - 2$ contains a right arrow, and
- if $\delta = +1$ then position $i + 2$ contains a left arrow.

These conditions guarantee that the surrounding arrows correctly point to the Turing machine head before and after the move. If any of these conditions is not satisfied then instead of the regular move by the Turing machine, the symbol '+' or '-' is swapped so that the direction of the simulation is reversed.

It follows from the reversibility of M that A is a reversible CA. If M is not periodic then it has a non-periodic configuration $(s, c) \in S \times \Sigma^{\mathbb{Z}}$. Clearly A then has a non-periodic configuration whose first tract reads c and the second track reads ${}^{\omega} \rightarrow s \leftarrow^{\omega}$.

Conversely, assume that M is periodic, with period p. Let $c \in Q^{\mathbb{Z}}$ be an arbitrary configuration of A. Configuration c is only changed around cells that contain a TM state, but there may be any number of such cells. However, due to the left and right arrows of the second track, different Turing machine simulations cannot interfere with each other: All activity is constrained within segments of the form $\rightarrow^n s \leftarrow^m$ where $n, m \in \mathbb{N} \cup \{\infty\}$ and $s \in S$. In each such segment the Turing machine either acts periodically with period p, or before time p reaches the end of the segment and reverses. As the inverse TM is also periodic with period p, it is clear that in the second case another reverse will take place and the action is periodic with period at most $2p$. Since all segments are periodic with period at most $2p$, the CA is periodic with period $(2p)!$.

The result now follows from Theorem 23. ■

A one-dimensional RCA is equicontinuous if and only if it is periodic, so we have

Corollary 29 *It is undecidable whether a given one-dimensional reversible CA is equicontinuous.*

3.4 Periodic Orbits

Every cellular automaton has periodic orbits so the existence of periodic orbits is trivial among cellular automata.

References

[1] C. B. Bennett, Logical reversibility of computation, IBM Journal of Research and Development 17 (6) (1973) 525–532.

[2] V. D. Blondel, J. Cassaigne, C. M. Nichitiu, On the presence of periodic configurations in turing machines and in counter machines, Theor. Comput. Sci. 289 (1) (2002) 573–590.

[3] P. Collins, van Schuppen. J. H., Observability of hybrid systems and turing machines, in: 43rd IEEE Conference on Decision and Control, vol. 1, IEEE Press, 2004.

[4] P. K. Hooper, The undecidability of the turing machine immortality problem, J. Symb. Log. 31 (2) (1966) 219–234.

[5] J. Kari, Theory of cellular automata: a survey, Theor. Comput. Sci. 334 (2005) 3–33.

[6] J. Kari, V. Lukkarila, Some undecidable dynamical properties for one-dimensional reversible cellular automata, in: A. Condon, D. Harel, J. N. Kok, A. Salomaa, E. Winfree (eds.), Algorithmic Bioprocesses, Springer-Verlag, 2008, *to appear*.

[7] J. Kari, N. Ollinger, Periodicity and immortality in reversible computing, in: E. Ochmański, J. Tyszkiewicz (eds.), Mathematical Foundations of Computer Science (MFCS'2008), vol. 5162 of Lecture Notes in Computer Science, Springer, Berlin, 2008, pp. 419–430.

[8] P. Kůrka, On topological dynamics of turing machines, Theor. Comput. Sci. 174 (1-2) (1997) 203–216.

[9] R. Landauer, Irreversibility and heat generation in the computing process, IBM Journal of Research and Development 5 (1961) 183–191.

[10] Y. Lecerf, Machines de turing réversibles, C. R. Acad. Sci. Paris 257 (1963) 2597–2600.

[11] M. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, Englewoods Cliffs, 1967.

[12] K. Morita, Universality of a reversible two-counter machine, Theor. Comput. Sci. 168 (2) (1996) 303–320.

[13] N. Ollinger, The Gnirut toolkit, software package `http://www.lif.univ-mrs.fr/~nollinge/rec/gnirut/`.

B.4 Playing with Conway's Problem

Version finale de [J2], un article écrit en collaboration avec E. Jeandel consacré à une démonstration de la non récursivité du commutant des langages rationnels par le biais de modèles de calcul *ad hoc* et de jeux. Cet article présente une démonstration plus informatique d'un résultat de M. Kunc [54] qui clôt un problème ouvert introduit par J. H. Conway [16] et remis au goût du jour par C. Choffrut *et al.* [J1].

Playing with Conway's Problem

Emmanuel Jeandel [a,b], Nicolas Ollinger [b,*]

[a] *LIP, École Normale Supérieure de Lyon, CNRS*
46 allée d'Italie, 69007 Lyon, France

[b] *LIF, Aix-Marseille Université, CNRS,*
39 rue Joliot-Curie, 13013 Marseille, France

Abstract

The centralizer of a language is the maximal language commuting with it. The question, raised by Conway in 1971, whether the centralizer of a rational language is always rational, recently received a lot of attention. In Kunc 2005, a strong negative answer to this problem was given by showing that even complete co-recursively enumerable centralizers exist for finite languages. Using a combinatorial game approach, we give here an incremental construction of rational languages embedding any recursive computation in their centralizers.

In 1999, Choffrut *et al.* [1] renewed an old problem raised by Conway [2] in 1971: given a rational language, does its centralizer — the maximal language commuting with it — have to be rational? The property is known to hold for some particular families of languages. In the case of codes, Ratoandramanana [3] showed in 1989 that it holds for prefix codes, raising a restriction of Conway's problem to codes which recently recieved a positive answer by Karhumäki *et al.* [4]. In the general case, until recently, the best known result, by Karhumäki and Petre [5], was that the centralizer of a recursive language has to be co-recursively enumerable. This property may also be considered as a particular case of results of Okhotin [6] concerning the computational power of systems of equations on languages. For a complete survey on Conway's problem, the reader may refer to [7–10]. In 2004, the community was thrilled by an announcement by Kunc [11] that a centralizer can actually be non-recursive. This announcement was followed by a conference communication [12] in 2005 showing that finite languages exist whose centralizers are complete for co-recursively enumerable languages [1]. It includes a sketch of the proof for the special case of rational languages. While simpler than the proof

* Corresponding author.
Email address: Nicolas.Ollinger@lif.univ-mrs.fr (Nicolas Ollinger).

[1] Since the writing of the present paper, a journal version appeared in [13]

for finite languages, this proof is still rather involved — mostly due to a direct construction of the language encoding a given Minsky machine.

In this paper we propose another proof of the existence of rational languages with non-recursive centralizers. The key arguments of the proof come from a careful study of the first example in Kunc [12] leading to the core constructions of our proof: *checking* and *flooding*. Our approach significantly differs for two reasons. First, a combinatorial game point of view is taken through the whole proof. Games are convenient tools to embed a dynamical process like a computation into a static object like a fix-point. Using this point of view, a computation can be transformed incrementally into a centralizer by transforming winning strategies from one game to another more specialized game. Secondly, the construction of the language embedding a particular computation is incremental — explicitly explaining how to compile any program into a language so that its centralizer corresponds to the computation. Whereas the final proof is by no way shorter than Kunc original proof, cutting the construction into locally independent propositions improves its readability. Our proof also uses Post tag systems instead of Minsky machines as Post tag systems are in a way closer to centralizers.

In this paper, the letters Σ and Γ denote *finite alphabets*. The set of finite words over an alphabet Σ is denoted by $\Sigma^\star$, the *empty word* by ε, the *catenation* of two words x and y by xy and the length of $x \in \Sigma^\star$ by $|x|$. A word x is a *prefix* (resp. *suffix*) of a word y, if there exists a word $z \in \Sigma^\star$ such that $xz = y$ (resp. $y = zx$); this word z is unique and is denoted as $x^{-1}y$ (resp. yx^{-1}). A word x is a *subword* of a word y if there exist two words $z, z' \in \Sigma^\star$ such that $zxz' = y$. A *language* over Σ is a subset of $\Sigma^\star$. The *product* XY of two languages X and Y is the language $\{xy : x \in X, y \in Y\}$. The language of prefixes (resp. suffixes) of a language X, denoted as $\mathrm{Pref}(X)$ (resp. $\mathrm{Suff}(X)$), is the set of all prefixes (resp. suffixes) of words in X. The language of subwords of a language X, denoted as $\mathrm{Sub}(X)$, is the set of all subwords of words in X. The language $X^{-1}Y$ is the language $\{z : \exists x \in X, \exists y \in Y, y = xz\}$. The language YX^{-1} is the language $\{z : \exists x \in X, \exists y \in Y, y = zx\}$.

Two languages X and Y *commute* if the equation $XY = YX$ is satisfied. The set of languages that commute with a given language X is closed under infinite union. Thus it admits a unique maximal element for inclusion called the *centralizer* of X, denoted by $\mathcal{C}(X)$. The centralizer of X always contains $X^\star$. Moreover, if X contains the empty word then its centralizer is equal to $\Sigma^\star$. Otherwise, it is contained in $\mathrm{Pref}(X^\star) \cap \mathrm{Suff}(X^\star)$.

Conway's Problem Is $\mathcal{C}(X)$ rational if X is rational?

The paper is constructed as follows. In section 1, a particular family of games called cutenation games are introduced. These games can be viewed as an

extension of tag systems with states, languages constraints on states and the ability to cut and catenate on both sides of the word. In section 2, tag systems are encoded into games verifying some regularity properties. In section 3, these games are recursively transformed into games with only two states. In section 4, language constraints on both states are removed. In section 5, every parts are glued together to obtain the main result.

1 Cutenation games

In this section cutenation [2] games are introduced and their relations with centralizers are explained before sketching the proof of existence of rational languages with non-recursive centralizers.

1.1 Definition

A *cutenation game* is a tuple $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ where $\mathfrak{A}$ and $\mathfrak{B}$ are finite, both $(\mathfrak{A}, \mathfrak{B}, L)$ and $(\mathfrak{A}, \mathfrak{B}, R)$ are bipartite graphs whose edges are tagged with words on Σ (*i.e.* $L, R \subseteq \mathfrak{A} \times \mathfrak{B} \times \Sigma^\star$) and the mappings $V_A : \mathfrak{A} \to \mathrm{Rat}(\Sigma^\star)$ and $V_B : \mathfrak{B} \to \mathrm{Rat}(\Sigma^\star)$ constraint the positions. Given such a game, an *A-configuration* $(\mathfrak{a}, x) \in \mathfrak{A} \times \Sigma^\star$ verifies $x \in V_A(\mathfrak{a})$. A pair $(\mathfrak{a}, x) \in \mathfrak{A} \times \Sigma^\star$ might not be a valid position of the game. Symmetrically a *B-configuration* $(\mathfrak{b}, y) \in \mathfrak{B} \times \Sigma^\star$ verifies $y \in V_B(\mathfrak{b})$.

Remark. In this paper we will only consider connected cutenation games, that is cutenation games $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ for which the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L \cup R)$ is connected.

Notation. We will depict L and R by a graph where $\mathfrak{A}$-vertices are represented by black points, $\mathfrak{B}$-vertices are represented by white points, L-edges are represented by plain edges and R-edges are represented by dashed edges (for clarity ε tags will be omitted).

Example. A sample cutenation game, omitting V_A and V_B, is depicted on Fig. 1 where $\mathfrak{A} = \{\alpha, \beta, \gamma, \delta\}$, $\mathfrak{B} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, $L = \{(\alpha, \mathrm{b}, \varepsilon), (\alpha, \mathrm{c}, \varepsilon), (\beta, \mathrm{a}, ab)\}$ and $R = \{(\alpha, \mathrm{a}, \varepsilon), (\beta, \mathrm{c}, \varepsilon), (\gamma, \mathrm{c}, \varepsilon), (\delta, \mathrm{b}, baa), (\delta, \mathrm{c}, \varepsilon)\}$.

A cutenation game is played as an iterated two-player combinatorial game where the set of A-configurations is the set of positions of the player A and

[2] *cutenation* is a free contraction of both words *cut* and *catenation*.

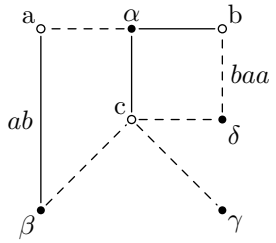


Fig. 1. graphical representation of a simple cutenation game

the set of B-configurations is the set of positions of the player B. A move of the player A, from an A-configuration $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, is a catenation:

- either an l-move $(\mathfrak{a}, x) \vdash_{A,l} (\mathfrak{b}, zx)$ such that $y = zx$ and $(\mathfrak{a}, \mathfrak{b}, z) \in L$;
- or an r-move $(\mathfrak{a}, x) \vdash_{A,r} (\mathfrak{b}, xz)$ such that $y = xz$ and $(\mathfrak{a}, \mathfrak{b}, z) \in R$.

Symmetrically, a move of the player B, from a B-configuration $(\mathfrak{b}, y)$ to an A-configuration $(\mathfrak{a}, x)$, is a cut:

- either an l-move $(\mathfrak{b}, zx) \vdash_{B,l} (\mathfrak{a}, x)$ such that $y = zx$ and $(\mathfrak{a}, \mathfrak{b}, z) \in L$;
- or an r-move $(\mathfrak{b}, xz) \vdash_{B,r} (\mathfrak{a}, x)$ such that $y = xz$ and $(\mathfrak{a}, \mathfrak{b}, z) \in R$.

A round of the game starts from an A-configuration $(\mathfrak{a}, x)$ and consists first of a move of the player A from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, then of a move of the player B from $(\mathfrak{b}, y)$ to an A-configuration $(\mathfrak{a}', x')$. Furthermore, if A plays an l-move then B must play an r-move and symmetrically if A plays an r-move then B must play an l-move. If a player cannot move then the player looses. The next round will start from $(\mathfrak{a}', x')$. If the game lasts forever then the player B wins.

Example. For the cutenation game of Fig. 1, this is a valid sequence of consecutive rounds, assuming $V_A(\mathfrak{a})$ and $V_B(\mathfrak{b})$ always equal to $\Sigma^\star$:

$$\begin{aligned}
&1.\ (\beta, aa) \vdash_{A,l} (\mathrm{a}, abaa) \vdash_{B,r} (\alpha, abaa)\ ;\\
&2.\ (\alpha, abaa) \vdash_{A,l} (\mathrm{b}, abaa) \vdash_{B,r} (\delta, a)\ ;\\
&3.\ (\delta, a) \vdash_{A,r} (\mathrm{c}, a) \vdash_{B,l} (\alpha, a)\ .
\end{aligned}$$

Notice that such a game can be played without memory. Thus a *strategy* for the player A in this game is simply a mapping from A-configurations to valid moves from the given configuration. The strategy is winning for a given configuration if, when the player A plays according to the strategy, whatever moves the player B decide to play, the player A wins the game. Strategies and winning strategies for the player B are defined symmetrically. A configuration

is called a winning position for a given player if there exists a winning strategy for this configuration for this player. The following classical result holds for these games, we give here a sketch of a proof.

Lemma 1 *Starting from an A-configuration* $(\mathfrak{a}, x)$ *either the player A has a winning strategy or the player B has a winning strategy.*

PROOF. Let $(\mathfrak{a}, x)$ be an A-configuration for which neither the player A nor the player B has a winning strategy. If every move from the player A starting from $(\mathfrak{a}, x)$ would lead to a B-configuration from which the player B could move to an A-configuration on which the player B has a winning strategy then the position $(\mathfrak{a}, x)$ would be winning for the player B. Thus, the player A has a valid move from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$ from which the player B can move either to A-configurations on which the player A has a winning strategy or to A-configurations on which neither the player A nor the player B has a winning strategy. On such configurations the best moves from both the player A and the player B would lead to an infinite run. By the rules, the player B would win which implies that the player B has a winning strategy starting from $(\mathfrak{a}, x)$. □

1.2 Languages and centralizers

Given a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ and an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is the set of words $x \in \Sigma^\star$ such that the configuration $(\mathfrak{a}, x)$ admits a winning strategy for the player B.

In the special case where both $\mathfrak{A}$ and $\mathfrak{B}$ are finite and L, R, V_A and V_B are recursive, given an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is co-recursively enumerable. It follows from the fact that one can exhaustively search a winning strategy for the player A as a finite one exists — the player B only has finitely many valid moves starting from a B-configuration, as one word has finitely many subwords.

The centralizer $\mathcal{C}(X)$ of a given language X can be expressed as the language $\mathfrak{L}$ associated with the unique element of $\mathfrak{A}$ of the cutenation game where both $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, L and R are both equal to the language X, and both $V_A(\mathfrak{a}) = \Sigma^\star$ and $V_B(\mathfrak{b}) = \Sigma^\star$. In the following, we call such a game a commutation game. For the sake of readability, when manipulating cutenation game where $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, we will manipulate L, R, V_A and V_B as subsets of $\Sigma^\star$ and denote the language associated with the game as $\mathfrak{L}$. For the same reasons A-configurations and B-configurations will be considered as elements of $\Sigma^\star$.

In order to prove the main result of this paper, we will proceed through the following steps. First, we restrict ourselves to a specific subset of special cutenation games. Then, we show how to recursively encode co-recursively enumerable languages into the language of such a game. After that we proceed to the core of the proof and explain how to transform such special cutenation game into a commutation game. During this transformation, the language associated with any element of $\mathfrak{A}$ is recursively encoded into the language associated with the commutation game of a rational language.

2 Encoding Post Tag Systems

In order to encode every co-recursively enumerable language into the language associated with a commutation game, the family of cutenation games is first restricted to games with special properties that will allow further reductions; then Post tag systems are encoded into games verifying these particular properties.

2.1 Restraining Cutenation Games

The following special kinds of cutenation games will be used in the proof. The main reason to enforce these properties is to enable the later encoding of both $\mathfrak{A}$ and $\mathfrak{B}$ into L, R, V_A and V_B.

Unfairness. A cutenation game is *unfair* if the player A has no constraint. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ over the alphabet Σ is *unfair* if for all $\mathfrak{b} \in \mathfrak{B}$, $V_B(\mathfrak{b}) = \Sigma^\star$.

Rootedness. A cutenation game is *rooted* if the player A can catenate non-empty words on the left (respectively on the right) from at most one position called the left root (respectively the right root). More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *rooted* if there exists a left root $\mathfrak{a}_L \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_L$ and there exists a right root $\mathfrak{a}_R \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_R$.

Oscillation. A cutenation game is *oscillating* if the player A is enforced to oscillate at each round between l-moves and r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oscillating* if the set $\mathfrak{A}$ can be split into two disjoint sets $\mathfrak{A}_L$ and $\mathfrak{A}_R$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ necessarily $\mathfrak{a} \in \mathfrak{A}_L$ and for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ necessarily $\mathfrak{a} \in \mathfrak{A}_R$.

SEPARATION. A cutenation game is *separated* if positions can be viewed as pairs of left and right positions, a left position being only affected by l-moves and a right position only by r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *separated* if there exist two sets S_L and S_R such that $\mathfrak{A} \cup \mathfrak{B} \subseteq S_L \times S_R$ and both L and R satisfy the following requirements. For every move $((s,t),(s',t'),x) \in L$ only the left part is modified, so $t = t'$. Moreover, for every $t'' \in S_R$ such that $(s,t'') \in \mathfrak{A}$ necessarily $(s',t'') \in \mathfrak{B}$ and the move $((s,t''),(s',t''),x)$ must be in L. Symmetrically, for every move $((s,t),(s',t'),x) \in R$ only the right part is modified, so $s = s'$. Moreover, for every $s'' \in S_L$ such that $(s'',t) \in \mathfrak{A}$ necessarily $(s'',t') \in \mathfrak{B}$ and the move $((s'',t),(s'',t'),x)$ must be in R.

ORIENTATION. A cutenation game is *oriented* if it is both separated and oscillating and if its left and right positions can be ordered into minimal and maximal positions, a move changing the corresponding position from minimal to maximal. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oriented* if it is both separated and oscillating and if the set S_L, respectively S_R, can be split into two disjoint sets S_L^- and S_L^+, respectively S_R^- and S_R^+, such that $\mathfrak{A}_L \subseteq S_L^- \times S_R^+$, $\mathfrak{A}_R \subseteq S_L^+ \times S_R^-$, and $\mathfrak{B} \subseteq S_L^+ \times S_R^+$.

Lemma 2 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game. Let ν_L, symmetrically ν_R, be the function mapping an element of $\mathfrak{A} \cup \mathfrak{B}$ to its connected component in the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L)$, symmetrically $(\mathfrak{A}, \mathfrak{B}, R)$. If the mapping $\nu : \mathfrak{a} \mapsto (\nu_R(\mathfrak{a}), \nu_L(\mathfrak{a}))$ is injective then the given oscillating cutenation game can be considered, up to the isomorphism ν, as a separated oscillating cutenation game where $S_L = \nu_R(\mathfrak{A} \cup \mathfrak{B})$ and $S_R = \nu_L(\mathfrak{A} \cup \mathfrak{B})$.*

PROOF. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game satisfying the hypothesis. Let $(\mathfrak{a}, \mathfrak{b}, x)$ be in L and let both $(s,t) = \nu(\mathfrak{a})$ and $(s',t') = \nu(\mathfrak{b})$. By definition of ν_L, as $\mathfrak{a}$ and $\mathfrak{b}$ are connected by L then $\nu_L(\mathfrak{a}) = \nu_L(\mathfrak{b})$ thus $t = t'$. Moreover, as the game is oscillating $\mathfrak{a} \in \mathfrak{A}_L$. Let $t'' \in S_R$ be such that $(s,t'') = \nu(\mathfrak{a}')$ for some $\mathfrak{a}' \in \mathfrak{A}$. By definition of ν_R this means that $\mathfrak{a}$ and $\mathfrak{a}'$ are connected by R. As $\mathfrak{a} \in \mathfrak{A}_L$ it implies that $\mathfrak{a} = \mathfrak{a}'$. A symmetrical reasoning applies to R. Therefore, the game is, up to the isomorphism ν, separated. □

Lemma 3 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a separated oscillating cutenation game obtained by Lemma 2. Such a game is oriented.*

PROOF. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a separated oscillating cutenation game obtained by Lemma 2. Let S_L^+ be defined as $\{s : \exists t \in S_R, (s,t) \in \mathfrak{B}\}$ and $S_L^- = S_L \setminus S_L^+$. Symmetrically, let S_R^+ be defined as $\{t : \exists s \in S_L, (s,t) \in \mathfrak{B}\}$ and $S_R^- = S_R \setminus S_R^+$. By construction, the inclusion $\mathfrak{B} \subseteq S_L^+ \times S_R^+$ holds. Let (s,t)

be in $\mathfrak{A}_L$. As the cutenation game is connected there is at least one move in L involving (s,t) thus $t \in S_R^+$. Assume that $s \in S_L^+$. This means that there exists some $t' \in S_R^+$ such that $(s,t') \in \mathfrak{B}$. By definition of ν_R necessarily (s,t) and (s,t') are connected by R. As the game is oscillating it implies that $t = t'$, but $\mathfrak{A}$ and $\mathfrak{B}$ are disjoint. This is a contradiction, therefore s must be in S_L^-. Symmetrically, the same holds for $\mathfrak{A}_R$. □

2.2 Post Tag Systems

A *Post tag system* $\mathcal{P}$ is a triple (Σ, k, φ) where Σ is a finite alphabet, k is a positive integer and φ is a mapping from Σ^k to $\Sigma^\star$. A configuration of the system is a word from $\Sigma^\star$. For all y in $\Sigma^\star$ and i in Σ^k, the configuration iy evolves into the configuration $y\varphi(i)$. The computation stops when no further evolution is possible, *i.e.* when the length of the word is less than k. The language $L_{\mathcal{P}}$ associated with the Post tag system is the set of words for which the evolution eventually stops. Post tag systems are universal in the sense that one can recursively encode any recursively enumerable language into their languages. For more details about tag systems and their computational power, the reader might consult Minsky [14].

Proposition 4 *Let $\mathcal{P}$ be a Post tag system over the alphabet Σ. There exists an unfair rooted oriented cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ over the same alphabet such that, for some distinguished element $\mathfrak{a} \in \mathfrak{A}$, the languages $\mathfrak{L}(\mathfrak{a})$ and $\Sigma^\star \setminus L_{\mathcal{P}}$ are equal.*

PROOF. Let $\mathcal{P}$ be a Post tag system (Σ, k, φ). The tag system will be encoded as a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ where

$$\mathfrak{A} = \{\alpha, \eta\} \cup \bigcup_{i \in \Sigma^k} \{\beta_i, \gamma_i, \delta_i, \zeta_i\},$$
$$\mathfrak{B} = \{\mathrm{a}\} \cup \bigcup_{i \in \Sigma^k} \{\mathrm{b}_i, \mathrm{c}_i, \mathrm{d}_i\}$$

and the relations L and R are depicted on Fig. 2.

The constraints V_A are defined as follows: $V_A(\alpha) = \Sigma^\star$, $V_A(\eta) = \Sigma^\star$, and for all i in Σ^k:

$$V_A(\beta_i) = \left(\Sigma^k \setminus \{i\}\right) \Sigma^\star \varphi(i),$$
$$V_A(\gamma_i) = i\Sigma^\star \varphi(i),$$
$$V_A(\delta_i) = i\Sigma^\star \varphi(i),$$
$$V_A(\zeta_i) = \Sigma^\star \setminus i\Sigma^\star \varphi(i).$$

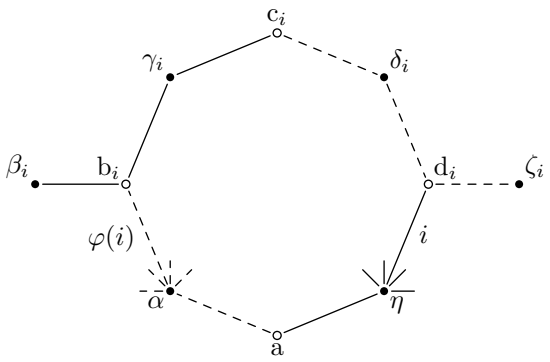


Fig. 2. the rooted oriented cutenation game of a Post system

This game is unfair and rooted, the roots being α and η. Moreover, it is oscillating and fulfills the requirements of Lemma 2 thus by Lemma 3 it is oriented. It remains to prove that $\mathfrak{L}(\alpha)$ equals $\Sigma^\star \setminus L_{\mathcal{P}}$.

Let x be a word in $L_{\mathcal{P}}$. A winning strategy for the player A starting from the A-configuration (α, x) is to follow the computation steps of the Post tag system. If a transition of the tag system exists starting from x then x can be written as iy with $i \in \Sigma^k$. Going through the states $\mathrm{b_i}$, $\mathrm{c_i}$, $\mathrm{d_i}$ and a, the player A will force the player B to go to the A-configuation $(\alpha, y\varphi(i))$. If no transition of the tag system exists starting from x, this means that $|x|$ is less than k, the player A moves to $\mathrm{b_i}$ for any $i \in \Sigma^k$. The player B has no valid move. The player A wins. Therefore, the player A has a winning strategy starting from (α, x) with $x \in L_{\mathcal{P}}$.

Let x be a word in $\Sigma^\star \setminus L_{\mathcal{P}}$. A winning strategy for the player B starting from the A-configuration (α, x) works as follows. In this game the player B has no choice so his strategy is to play when he can. The only possibility for the player B to have no valid move is to play from some position $\mathrm{b_i}$ obtained from the position α with a word of length less than k. Observe that the only possible sequences of moves going from a configuration (α, y) to a configuration (α, z) imply that in the tag system there is a valid sequence of forward and backward transitions from y to z. Thus, as in the tag system x has an infinite sequence of valid forward transitions, the position (α, x) is winning for the player B. Therefore, the player B has a winning strategy starting from (α, x) with $x \in \Sigma^\star \setminus L_{\mathcal{P}}$. □

3 Removing states

In order to transform unfair rooted oriented cutenation games into commutation games, the first step is to transform the state sets $\mathfrak{A}$ and $\mathfrak{B}$ into singletons and to ensure that $L = R$. This is done by choosing a proper encoding of every configuration $((s, t), x)$ into a proper word $\langle s, x, t\rangle$.

3.1 Encoding states

Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an unfair rooted oriented cutenation game. We encode each configuration $((s,t),x)$ into a proper word $\langle s,x,t\rangle$ using the following encoding.

Let m be the size of S_L^- and Γ_L^- be an alphabet of $m-1$ ordered new letters $\{\alpha_1,\dots,\alpha_{m-1}\}$. Let ρ map S_L^- into $\{0,1,\dots,m-1\}$ so that the left coordinate of the left root is mapped into 0. Let φ_L^- map $s \in S_L^-$ into the word $\alpha_{\rho(s)}\cdots\alpha_2\alpha_1$ of length $\rho(s)$. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^-(s)$ when $s \in S_L^-$. Let n be the size of S_L^+ and Γ_L^+ be an alphabet of n ordered new letters $\{\beta_1,\dots,\beta_n\}$. Let σ map S_L^+ into $\{1,\dots,n\}$. Let φ_L^+ map $s \in S_L^+$ into the word $\beta_{\sigma(s)}\alpha_{m-1}\cdots\alpha_1$ of length m. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^+(s)$ when $s \in S_L^+$. For each pair of states $(s,s') \in S_L^- \times S_L^+$ define $\phi_L(s,s')$ as $\varphi_L^+(s')\varphi_L^-(s)^{-1}$, which is $\beta_{\sigma(s')}\alpha_{m-1}\cdots\alpha_{\rho(s)+1}$. Notice that $\phi_L(s,s') \in \Gamma_L^+\left(\Gamma_L^-\right)^\star$.

Symmetrically, let m' be the size of S_R^- and Γ_R^- be an alphabet of $m'-1$ ordered new letters $\{\gamma_1,\dots,\gamma_{m'-1}\}$. Let ρ' map S_R^- into $\{0,1,\dots,m'-1\}$ so that the right coordinate of the right root is mapped into 0. Let φ_R^- map $t \in S_R^-$ into the word $\gamma_1\gamma_2\cdots\gamma_{\rho'(t)}$ of length $\rho'(t)$. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^-(t)$ when $t \in S_R^-$. Let n' be the size of S_R^+ and Γ_R^+ be an alphabet of n' ordered new letters $\{\delta_1,\dots,\delta_{n'}\}$. Let σ' map S_R^+ into $\{1,\dots,n'\}$. Let φ_R^+ map $t \in S_R^+$ into the word $\gamma_1\cdots\gamma_{m'-1}\delta_{\sigma'(t)}$ of length m'. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^+(t)$ when $t \in S_R^+$. For each pair of states $(t,t') \in S_R^- \times S_R^+$ define $\phi_R(t,t')$ as $\varphi_R^-(t)^{-1}\varphi_R^+(t')$, which is $\gamma_{\rho'(t)+1}\cdots\gamma_{m'-1}\delta_{\sigma'(t')}$. Notice that $\phi_R(t,t') \in \left(\Gamma_R^-\right)^\star\Gamma_R^+$.

Let τ_L and τ_R be the two morphisms from $\Sigma^\star$ to $(\Sigma\cup\{o\})^\star$, where o is a new letter, defined for each letter $a \in \Sigma$ by $\tau_L(a) = oa$ and $\tau_R(a) = ao$. For each word $x \in \Sigma^\star$ define $\tau(x)$ as $\tau_L(x)o$, which is equal to $o\tau_R(x)$. These morphisms will be used to encode configurations of the game with two goals in mind: first of all, ensure that no word is encoded as the empty word; secondly ensure that each encoded word has an odd length.

A configuration $((s,t),x) \in (S_L \times S_R)\times\Sigma^\star$ of the game will be encoded by the word $\varphi_L(s)\tau(x)\varphi_R(t)$ denoted as $\langle s,x,t\rangle$. The set L will be encoded using the mapping ψ_L defined by $\psi_L((s,t),(s',t),x) = \phi_L(s,s')\tau_L(x)$. Symmetrically, the set R will be encoded using the mapping ψ_R defined by $\psi_R((s,t),(s,t'),y) = \tau_R(y)\phi_R(t,t')$.

Remark. To summarize, $\mathfrak{A}_L$-configurations are encoded by words in the language $\mathrm{Suff}(\alpha_{m-1}\cdots\alpha_1)\,(o\Sigma)^\star o\gamma_1\cdots\gamma_{m'-1}\Gamma_R^+$, symmetrically $\mathfrak{A}_R$-configurations

are encoded by words in $\Gamma_L^+\alpha_{m-1}\cdots\alpha_1\,(o\Sigma)^\star o\,\mathrm{Pref}(\gamma_1\cdots\gamma_{m'-1})$, and finally, $\mathfrak{B}$-configurations are encoded by words in $\Gamma_L^+\alpha_{m-1}\cdots\alpha_1\,(o\Sigma)^\star o\gamma_1\cdots\gamma_{m'-1}\Gamma_R^+$.

Proposition 5 *Let $(\mathfrak{A},\mathfrak{B},L,R,V_A,\Sigma^\star)$ be an unfair rooted oriented cutenation game. Let V'_A be the set $\{\langle s,x,t\rangle : (s,t)\in\mathfrak{A}, x\in V_A((s,t))\}$ and V'_B the set $\mathrm{Sub}\left(\left\{\langle s,x,t\rangle : x\in\Sigma^\star,(s,t)\in S_L^+\times S_R^+\right\}\right)$. Let $((s,t),x)$ be an A-configuration of the game. There exists a valid l-move for the player A in the cutenation game $(\{\mathfrak{a}\},\{\mathfrak{b}\},\psi_L(L),\psi_R(R),V'_A,V'_B)$ from the configuration $\langle s,x,t\rangle$ to a configuration w if and only if $w=\langle s',y,t'\rangle$ for some s', y, t' and the l-move from $((s,t),x)$ to $((s',t'),y)$ is valid for the player A in the first game. The same holds for r-moves and B-configurations.*

PROOF. Let $(\mathfrak{A},\mathfrak{B},L,R,V_A,\Sigma^\star)$ be an unfair rooted oriented cutenation game. Let $((s,t),x)$ be a configuration of the game.

Let $((s',t'),y)$ be a configuration of the game such that a move from $((s,t),x)$ to $((s',t'),y)$ is valid. Let $w=\langle s',y,t'\rangle$. If the move is an l-move for the player A then $t=t'$ and $((s,t),(s',t),z)\in L$ where $y=zx$, and thus $\phi_L(s,s')\tau_L(z)\in\psi_L(L)$. To prove that this move is a valid l-move in the new game, it is sufficient to show that $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\langle s',y,t'\rangle$. If (s,t) is the left root then $\varphi_L(s)=\varepsilon$ and $\phi(s,s')=\varphi_L(s')$ thus $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\varphi_L(s')\tau(zx)\varphi_R(t)$. If (s,t) is not the left root then $z=\varepsilon$, and therefore $\phi_L(s,s')\tau_L(z)\,\langle s,x,t\rangle=\varphi_L(s')\tau(x)\varphi_R(t)$ as $\phi_L(s,s')\varphi_L(s)=\varphi_L(s')$. Therefore, if the move is a valid l-move for the player A in the original game then it is a valid l-move for the player A in the new game. The three other cases work on the same principle (do not forget to check with V_A in the case of a move for the player B).

Let w be a word such that there is a valid move for the player A in the new game from $\langle s,x,t\rangle$ to w where $(s,t)\in\mathfrak{A}$. If it is an l-move, there exist some s', s'' and z such that $\phi_L(s',s'')\tau_L(z)\in\psi_L(L)$ and $w=\phi_L(s',s'')\tau_L(z)\,\langle s,x,t\rangle$. As $w\in V'_B$ and both $s''\in S_L^+$ and $t\in S_R^+$ then $w=\langle s'',y,t\rangle$ for some $y\in\Sigma^\star$. This implies that $s=s'$ and $y=zx$. To prove that there is a valid l-move in the original game for the player A from the configuration $((s,t),x)$ to the configuration $((s'',t),zx)$ it is sufficient to show that $(s'',t)\in\mathfrak{B}$ and $((s,t),(s'',t),z)\in L$. As $\phi_L(s,s'')\tau_L(z)\in\psi_L(L)$ there exists some t' such that $((s,t'),(s'',t'),z)\in L$. As the original game is separated and both $(s,t)\in\mathfrak{A}$ and $(s,t')\in\mathfrak{A}$ then $(s'',t)\in\mathfrak{B}$ and $((s,t),(s'',t),z)\in L$. The case of an r-move for the player A works symmetrically.

Let w be a word such that there is a valid move for the player B in the new game from $\langle s,x,t\rangle$ to w where $(s,t)\in\mathfrak{B}$. If it is an l-move then there exists some s', s'' and z such that $\phi_L(s',s'')\tau_L(z)\in\psi_L(L)$ and $\phi_L(s',s'')\tau_L(z)w=\langle s,x,t\rangle$. As $w\in V'_A$ and both $s\in S_L^+$ and $t\in S_R^+$ then $s''=s$ and $w=\langle s',y,t\rangle$

where $(s', t) \in \mathfrak{A}$ and $y \in V_A((s', t))$ is such that $zy = x$. To prove that there is a valid l-move in the original game for the player B from the configuration $((s, t), zy)$ to the configuration $((s', t), y)$ it is sufficient to show that $(s', t) \in \mathfrak{A}_L$ and $((s', t), (s, t), z) \in L$. As $\phi_L(s', s)\tau_L(z) \in \psi_L(L)$ there exists some t' such that $((s', t'), (s, t'), z) \in L$. As the original game is separated and both $(s', t) \in \mathfrak{A}$ and $(s', t') \in \mathfrak{A}$ then $((s', t), (s, t), z) \in L$. The case of r-move for the player B works symmetrically. □

3.2 Enforcing symmetry

In a commutation game both sets of left moves L and right moves R are equal. If the sets V_A and V_B bring enough constraints to the game, both L and R can be replaced by $L \cup R$ to enforce this symmetry.

Proposition 6 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, L, R, V_A, V_B)$ be a cutenation game. Let X be the language $L \cup R$. If the two sets $RV_A \cap V_B$ and $V_AL \cap V_B$ are empty then the valid moves, both for the player A and the player B, are the same in the given game and in the cutenation game $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$.*

PROOF. Every move in the original game is allowed in the new game. Conversely, let $x \vdash_{A,l} y$ be a valid l-move for the player A in the new game. There exists $z \in X$ such that $y = zx$ so $y \in XV_A \cap V_B$. As $RV_A \cap V_B$ is empty, then $z \in L$ and the move is also valid in the original game. The three remaining cases are similar using the three other empty sets (use $V_AL \cap V_B$ for $\vdash_{A,r}$, use $R^{-1}V_B \cap V_A$ for $\vdash_{B,l}$, and use $V_BL^{-1} \cap V_A$ for $\vdash_{B,r}$). □

The construction to enforce symmetry can be applied directly after encoding the states as the new encoding verifies the required hypothesis.

Lemma 7 *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be some unfair rooted oriented cutenation game. Let V'_A and V'_B be defined as in Prop. 5. Let X be the set $\psi_L(L) \cup \psi_R(R)$. The valid moves for both the player A and the player B are the same in both games $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, \psi_L(L), \psi_R(R), V'_A, V'_B)$ and $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$.*

PROOF. By Prop. 6 it is sufficient to show that the four sets $\psi_R(R)V'_A \cap V'_B$, $V'_A\psi_L(L) \cap V'_B$, $\psi_R(R)^{-1}V'_B \cap V'_A$, and $V'_B\psi_L(L)^{-1} \cap V'_A$ are empty.

The language $\psi_R(R)V'_A$ does not intersect V'_B because every word of $\psi_R(R)V'_A$ contains an occurrence of a letter in Γ_R^+ before a letter o and this is never the case in V'_B. A symmetrical proof works for $V'_A\psi_L(L) \cap V'_B$.

The language $\psi_R(R)^{-1}V'_B$ does not intersect V'_A because $\psi_R(R)^{-1}V'_B$ only contains the empty word which is not in V'_A. The same holds for $V'_B\psi_L(L)^{-1}$. □

4 Removing constraints

In order to conclude the construction the constraints sets V_A and V_B must be removed. This part is the core of the proof. The construction proceeds in two steps : first we remove V_A through *checking*, then we remove V_B through *flooding*.

4.1 Checking

To remove V_A means to remove constraints on the positions at the end of a move from the player B. To ensure that the set of winning strategies of the player B does not grow, the idea is to allow the player A to challenge the player B if he plays outside of V_A by *checking* the validity of the move.

Proposition 8 *Let* $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ *be a cutenation game over the alphabet* Σ *with associated language* $\mathfrak{L}$*. Let* c *be a new letter not in* Σ*, let* $X' = X \cup cV_A^\star \cup V_A^\star c$ *and* $V'_B = V_B \cup cV_B \cup V_B c$*. Let* $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X', X', (\Sigma \cup \{c\})^\star, V'_B)$ *be the cutenation game over the alphabet* $\Sigma \cup \{c\}$ *with associated language* $\mathfrak{L}'$*. If the four sets* $X^{-1}X^{-1}V_B$*,* $V_BX^{-1}X^{-1}$*,* $((X^{-1}(V_AX \cap V_B)) \setminus V_A) \cap V_A^\star$*, and* $(((XV_A \cap V_B)X^{-1}) \setminus V_A) \cap V_A^\star$ *are empty and both inclusions* $X^{-1}V_B \subseteq V_B$ *and* $V_BX^{-1} \subseteq V_B$ *hold then* $\mathfrak{L}$ *is equal to* $\mathfrak{L}' \cap \Sigma^\star$*.*

PROOF. We prove that the player B has a winning strategy in the original game if and only if the player B has a winning strategy in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy as long as the player A keeps using moves that were valid in the original game. If the player A uses a new move from a position $x \in V_A$ then there are two possibilities:

- either he catenates a word of X, leading to a new position in $V'_B \setminus V_B$; this is impossible as all the new valid positions must contain the new letter c which does not appear in X;
- or he catenates a word of $X' \setminus X$ containing the new letter c, leading to a new valid position y which must be either in cV_B or in V_Bc; as $x \in V_A$ and as $X' \setminus X = cV_A^\star \cup V_A^\star c$, necessarily $y \in cV_A^\star \cup V_A^\star c$ and thus $y \in X'$.

A winning strategy for the player B starting from a position $y \in X'$ is simply to cut y completely thus leading to the empty word position. The empty word position is winning for the player B: when the player A catenates a word z, the player B just cuts z, coming back to the empty word. Therefore, the player B still has a winning strategy in the new game.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy as long as the player B keeps using moves that were valid in the original game. If the player B uses a new move from a position $x \in V_B$ then there are two possibilities:

- either he cuts a word of $X' \setminus X$ containing the new letter c; this is impossible as the new letter c does not appear in V_B;
- or he cuts a word of X, leading to a new valid position in $\Sigma^\star \setminus V_A$, more precisely in $((X^{-1}(V_A X \cap V_B)) \cup ((X V_A \cap V_B) X^{-1})) \setminus V_A$.

A winning strategy for the player A starting from a position y in the language $(X^{-1}(V_A X \cap V_B)) \setminus V_A$ is simply to catenate the word c on the right, leading to the valid position yc in $V_B c$. As $y \notin V_A^\star$ and $X^{-1}X^{-1}V_B = \emptyset$ the player B has no valid move starting from yc, thus the player A wins. By a symmetrical argument, the player A has a winning strategy starting from a position in $((X V_A \cap V_B) X^{-1}) \setminus V_A$. Therefore, the player A still has a winning strategy in the game. □

4.2 Flooding

To remove V_B means to remove constraints on the positions at the end of a move from the player A. To ensure that the set of winning strategies of the player A does not grow, the idea is to force every position outside of V_B to admit a winning strategy for the player B by *flooding* the language X with all words outside of V_B.

Proposition 9 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, \Sigma^\star, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathcal{L}$. If V_B is closed under subword then the centralizer $\mathcal{C}(X \cup \Sigma^\star \setminus V_B)$ is equal to $\mathcal{L}$.*

PROOF. We prove that the player B has a winning strategy in the cutenation game if and only if the player B has a winning strategy in the commutation game of $X \cup \Sigma^\star \setminus V_B$.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to his original strategy. As V_B

is closed under subword, starting from a word in V_B the player B cannot use a transition in $\Sigma^\star \setminus V_B$ to cut: the player B has exactly the same possible moves as in the original game. Therefore, the player A still has a winning strategy in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy as long as the player A keeps using moves that were valid in the original game. If the player A uses a new move then, as V_B is closed under subword, just after this move the new position is a word in $\Sigma^\star \setminus V_B$. A winning strategy for the player B starting from a word x in $\Sigma^\star \setminus V_B$ is simply to cut x completely thus accessing to the empty word position. The empty word position is winning for the player B: when the player A catenates a word y, the player B just cuts y, coming back to the empty word. Therefore, the player B still has a winning strategy in the new game. □

To combine both checking and flooding to remove the constraints on a game it is sufficient to ensure that the original constraints V_B are closed under subword.

Lemma 10 *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathfrak{L}$ such that V_B is closed under subword. Let c be a new letter not in Σ, let $X' = X \cup cV_A^\star \cup V_A^\star c$ and $V'_B = V_B \cup cV_B \cup V_B c$. If the four sets $X^{-1}X^{-1}V_B$, $V_B X^{-1}X^{-1}$, $((X^{-1}(V_A X \cap V_B)) \setminus V_A) \cap V_A^\star$, and $(((XV_A \cap V_B)X^{-1}) \setminus V_A) \cap V_A^\star$ are empty then $\mathfrak{L}$ is equal to*

$$\mathcal{C}(X \cup cV_A^\star \cup V_A^\star c \cup (\Sigma \cup \{c\})^\star \setminus (V_B \cup cV_B \cup V_B c)) \cap \Sigma^\star \quad .$$

PROOF. If V_B is closed under subword, so is $V'_B = V_B \cup cV_B \cup V_B c$. To conclude, combine both Prop. 8 and Prop. 9. □

5 Gluing all together

We can now conclude the proof of the main statement by combining the three parts of the construction together.

Theorem 11 *There exists a rational language X the centralizer $\mathcal{C}(X)$ of which is complete for co-recursively enumerable languages.*

PROOF. Let $\mathcal{P}$ be a Post tag sytem, the language of which is complete for recursively enumerable languages. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be the unfair

rooted oriented cutenation game obtained by Prop. 4 and such that for some $\mathfrak{a} \in \mathfrak{A}$ the equation $\mathfrak{L}(\mathfrak{a}) = \Sigma^\star \setminus L_\mathcal{P}$ holds. Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$ be the cutenation game with language $\mathfrak{L}$ obtained by Prop. 5 and Lemma 7 such that the equation $\mathfrak{L} \cap \{\langle s, x, t\rangle, x \in \Sigma^\star\} = \{\langle s, x, t\rangle, x \in \Sigma^\star \setminus L_\mathcal{P}\}$ holds for some $(s, t) \in \mathfrak{A}$. To combine this cutenation game with Lemma 10, as V'_B is closed under subword it is sufficient to show that the hypotheses of Prop. 8 are satisfied. More precisely, it is sufficient to show that the four sets $X^{-1}X^{-1}V'_B$, $V'_BX^{-1}X^{-1}$, $((X^{-1}(V'_AX \cap V'_B)) \setminus V'_A) \cap V'^\star_A$, and $(((XV'_A \cap V'_B)X^{-1}) \setminus V'_A) \cap V'^\star_A$ are empty and both inclusions $X^{-1}V'_B \subseteq V'_B$ and $V'_BX^{-1} \subseteq V'_B$ hold. Both inclusions hold because V'_B is closed under subword.

The set $X^{-1}X^{-1}V'_B$ is empty because first $\psi_R(R)^{-1}V'_B$ only contains the empty word and X does not contain the empty word, secondly because $\psi_L(L)^{-1}V'_B$ contains only the empty word and words which begin with a letter in $\Gamma^-_L \cup \{o\}$ and words in X never begin with such a letter. Symmetrically, the set $V'_BX^{-1}X^{-1}$ is empty.

The set $((X^{-1}(V'_AX \cap V'_B)) \setminus V'_A) \cap V'^\star_A$ is empty because all words in the language $X^{-1}(V'_AX \cap V'_B)$ contain exactly one occurence of a letter from $\Gamma^+_L \cup \Gamma^+_R$. Symmetrically, the set $(((XV'_A \cap V'_B)X^{-1}) \setminus V'_A) \cap V'^\star_A$ is empty.

Therefore, Lemma 10 can be applied and $\Sigma^\star \setminus L_\mathcal{P}$ can be recursively computed from the centralizer of the rational set $X \cup cV'^\star_A \cup V'^\star_Ac \cup (\Sigma' \cup \{c\})^\star \setminus (V'_B \cup cV'_B \cup V'_Bc)$. As a consequence, the centralizer of this rational language is complete for co-recursively enumerable languages. □

References

[1] C. Choffrut, J. Karhumäki, N. Ollinger, The commutation of finite sets: a challenging problem, Theoret. Comput. Sci. 273 (2002) 69–79.

[2] J. H. Conway, Regular Algebra and Finite Machines, Chapman Hall, 1971.

[3] B. Ratoandramanana, Codes et motifs, RAIRO Theor. Informat. 23 (1989) 425–444.

[4] J. Karhumäki, M. Latteux, I. Petre, The commutation with codes and ternary sets of words, Theoret. Comput. Sci. 340 (2005) 322–333.

[5] J. Karhumäki, I. Petre, Conway's problem for three-word sets, Theoret. Comput. Sci. 289 (2002) 705–725.

[6] A. Okhotin, Decision problems for language equations with boolean operations, in: Proc. of ICALP 2003, Vol. 2719 of LNCS, Springer, 2003, pp. 239–251.

[7] J. Karhumäki, Challenges of commutation: an advertisement, in: Proc. of FCT 2001, Vol. 2138 of LNCS, Springer, 2001, pp. 15–23.

[8] T. Harju, O. Ibarra, J. Karhumäki, A. Salomaa, Decision questions in semilinearity and commutation, J. Comput. Syst. Sci. 65 (2002) 278–294.

[9] I. Petre, Commutation problems on sets of words and formal power series, Ph.D. thesis, University of Turku (2002).

[10] J. Karhumäki, I. Petre, Two problems on commutation of languages, in: G. Rozenberg, A. Salomaa (Eds.), Current Trends in Theoretical Computer Science, World Scientific, 2004.

[11] M. Kunc, Regular solutions of language inequalities and well quasi-orders, in: Proc. of ICALP 2004, Vol. 3142 of LNCS, Springer, 2004, pp. 870–881.

[12] M. Kunc, The power of commuting with finite sets of words, in: Proc. of STACS 2005, Vol. 3404 of LNCS, Springer, 2005, pp. 569–580.

[13] M. Kunc, The power of commuting with finite sets of words, Theory of Computing Systems 40 (4) (2007) 521–551.

[14] M. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, 1967.